ACCEPTANCE SHEET

The special problem entitled "REND VISUALIZATION TOOL: An Automation and Simulation with Conversion System of four different notations of Regular Language " prepared and submitted by John Christopher C. Pasco in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

Vincent Peter C. Magboo, MD, MS
Adviser

EXAMINERS	APPROVED	DISAPPROVED
1. Gregorio B. Baes, Ph.D (Candidate)		
2. Avegail D. Carpio, MS (Candidate)		
3. Richard Bryann L. Chua, MS		
4. Aldrich Colin A. Co, MS (Candidate)		
5. Ma. Shiela A. Magboo, MS		
6. Geoffrey A. Solano, MS		

Date

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

Geoffrey A. Solano, MS
Unit Head
Mathematical and Computing Science Unit
Department of Physical Sciences
and Mathematics

Alex C. Gonzaga, Ph.D
Chair
Department of Physical Sciences
and Mathematics

Reynaldo H. Imperial, Ph.D

Dean
College of Arts and Sciences

ABSTRACT

E-Learning has been widely developed all over the world in efficiently utilizing technologies in acquiring knowledge. The REND Visualization Tool is a supplemental tool in teaching the basic concept of Automata Theory and Languages – the Regular Language. REND Visualization Tool is a standalone system that can convert representations of regular language namely DFA, NFA, ε-NFA and Regular Expression. Assessing the acquired knowledge is very important in enhancing one's ability. The system also includes tutorial and assessment for students to utilize. The system gives the user the privilege to save energy in performing manual computations in converting representation from one form to another. Also, a table used for computation is provided for details of computation.

Keywords: Automata Theory, E-Learning, DFA, NFA, Regular Expression, Finite Automaton, Regular Language

TABLE OF CONTENTS

I. INTRODUCTION

A. Background of the Study

A finite automaton has a set of states, and its "control" moves from state to state in response to external "inputs." Finite Automaton can either be deterministic (DFA) meaning that an automaton cannot be in more than one state at any one time or non-deterministic (NFA) meaning that it may be in several states at once. Extended non-deterministic automaton (ε-NFA) that has additional choice of making a transition from one state to another spontaneously, i.e. on the empty string as an "input", also accepts nothing but the regular languages.

A deterministic finite automaton is represented by a "five-tuple",

DFA = (Q, Σ, δ, q_0, F) where:

Q = finite set of states

Σ = input symbols

δ = transition function

q_0 = start state from set Q

F = set of final states and is a subset of Q

A non-deterministic finite automaton is represented essentially like a DFA:

NFA = (Q, Σ, δ, q_0, F) where:

Q is a finite set of states.

Σ is a finite set of input symbols.

q_0, a member of Q, is the start state.

F, a subset of Q, is the set of final (or accepting) states.

δ, the transition function is a function that takes a state in Q and an input symbol in Σ as arguments and returns a subset of Q. Notice that the only difference between an

NFA and a DFA is in the type of value that δ returns: a set of states in the case of an NFA and a single state in the case of DFA.

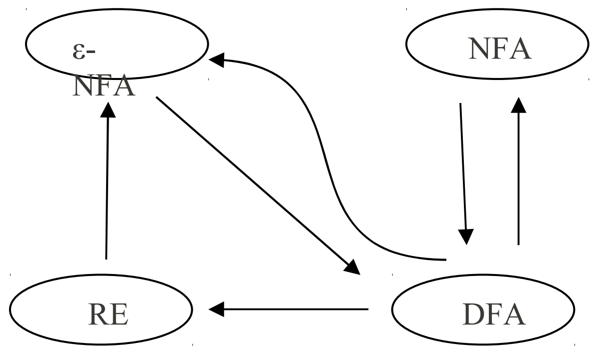
Note: In ε-NFA the δ takes the argument:

1. A state in Q, and

2. A member of Σ U {ε}, that is either an input symbol, or a symbol ε. [1]

Regular expressions define exactly the same languages that the various forms of automata describe: the regular languages. However, regular expressions offer something that automata do not: a declarative way to express the strings we want to accept by using some operators such as union (+), concatenation (. or no symbol at all) and closure (*), i.e. 01* + 10* denotes a language consisting of all strings that are either a single 0 followed by any number of 1's or a single 1 followed by any number of 0's.

Every finite automaton can be expressed by each notation and consequently, can be converted from one form to another. Also, a regular expression can represent exactly the same language an automaton describes. For every regular expression, there is an automaton equivalent to it and accordingly for every automaton, there is a regular expression equivalent to it. This representation describes the possible conversion:



Certain rules and steps are to be followed when converting from one form to another. [1]

B. Statement of the Problem

Automata and Language theory has many topics involving tedious computations that are prone to errors when done manually. Some of these topics are the finite automata and the regular expressions which are notations of regular languages. Student’s learning process of these topics involves the following problems:

a. The nature of the topic is naturally difficult.
b. Students have high percentage of committing errors due to long process of computations.
c. Students cannot easily assess themselves on how far they perceive the concepts.

There are commercially available systems that are somewhat a prototype of a system that performs tasks to solve the problems mentioned above. But this kind of systems encompasses also the much broader concepts of Automata and Language Theory. The Automation and Simulation with Conversion System of four different notations (DFA, NFA, ε-NFA, Regular Expression) of Regular Language only deals with the base concept of Automata – the Finite Automata. Thorough explanation can help the students, who are new to these concepts, to better understand and comprehend the ideas behind the more complex systems such as Context-Free, PDA, and Turing Machines. [2]

C. Objectives

The main objective of this study is to be able to create the Automation and Simulation with Conversion System of four different notations (DFA, NFA, ε-NFA, Regular Expression) of Regular Language, an automation and simulation system for finite automata and regular expressions, to help students in understanding concepts in the Theory of Automata and Language.

1. To automate the simulation of an input string given a DFA, NFA, ε-NFA, or Regular Expression and conversion of the following:

    a) From NFA to DFA

    b) From ε-NFA to DFA

    c) From RE to ε-NFA

    d) From DFA to RE

2. To create a tutorial/lessons about Regular Language and its representations.
3. To create an assessment examination of concepts and problem solving for users.

D. Significance of the Study

Automata and Languages Theory is highly theoretical field that requires a lot of effort in understanding the concepts. Its role is basically to provide us with a mathematical model for a computer or system. Some of the fields are the Finite Automaton and the Regular Expression that both described a specific regular language. Finite Automaton can be in the form of DFA, NFA and ε-NFA. Each form can be converted to another form. Regular Expression has an equivalent or can be converted to a finite automaton and vice versa. Computations and visualization in mind of these topics are very difficult to both in the part of the teachers and students

The automation and simulation of these concepts will not only make the tasks involve in tracing of input and conversion from one form to another exciting but will also make the works a lot easier and easy to understand. Also when the system is automated, the errors are assumed to be eliminated. In the system, visualization is no longer a problem with the aid of graphics and animation. The system can also serve as a supplemental tool in teaching of Automata Theory. Also, it can serve as a tutorial to those who want to explore the basis of Automata Theory- the Finite Automata by themselves.

E. Scope and Limitation

Automation and Simulation with Conversion System of four different notations (DFA, NFA, ε-NFA, Regular Expression) of Regular Language can trace if an input is part of a regular language depending on the component defined, based on the user's input. It is capable of converting one notation of a finite automaton to another. It is also capable of converting a regular expression to finite automaton and vice versa. On the other hand, the said system has the following bounds:

1. At most 10 states for a deterministic finite automaton (with system defined state names) only.
2. At most 5 states for a non-deterministic finite automaton(with system defined state names) only.
3. Regular Expression can have at most 4 terms
4. The symbols for input alphabet may come only from the binary alphabet

II. REVIEW OF RELATED LITERATURE

Automata and Languages Theory is an important part of the core of theoretical Computer Science. It is a convenient means of representing linguistic events. Studying the complexity of Automata and Languages Theory must first start with its heart or basis – the regular language. In fact in the past 20 years, a lot of research works have been devoted in exploring regular language. [3 - 8, 13 - 14]

Exploring regular languages is not enough to the researchers. They also devote some times in studying the behavior and possible development in the representations of regular language. The equivalence of the different representations like DFA, NFA, and regular expressions lead them to develop systems for conversion. Converting from one form to another must follow certain path and methods. A good example that has been studied is the conversion from Regular Expression to DFA using the compressed NFA. Another example is the opposite, the conversion of a finite automaton to a regular expression using the state elimination technique which is now widely used. They also developed a way to minimize the obtained NFA's and Regular Expressions in a certain condition. [5 - 7]

Regular Languages and their different representations like finite automaton and regular expressions are mainly used as lexical analyzer, communication protocol, and word search. But the applications of regular languages are smoothly widening in recent years. The finite automata are now used in security, images, and software modeling. [9 - 11]

One good example of application that is based in Automata is the INTEX by Silbertztein. It is an integrated Natural Language Processing toolbox based on finite automata with outputs or finite state transducers. It parses texts of numerous million words, and includes large-coverage dictionaries and grammars. Texts, Dictionaries and Grammars are internally represented as FST's. INTEX is used as lexical parser to produce the input of a syntactic parser. [8]

Another application is the Intrusion detection using DFA induction. Intrusion detection is used to detect some security threats. It is a key technology to self-healing systems to manage and prevent damage from security threats. Protecting web applications using intrusion detection is quite challenging because they are large and complex, and highly customized. Programmers of this kind of application create them with little security. DFA induction can be used to detect malicious web requests along with rules of variability among requests and heuristic for filtering and grouping anomalies. With this kind of setup, large possible attack can be prevented. [9]

Finite Automata are used in encoding and compression of images. For black and white images, for instance, they used the quad tree representations. The black points correspond to the ω-words defining the corresponding paths in the tree that lead to them. If the ω-language consisting of the set of all words is accepted by a deterministic finite automaton then the image is said to be encodable as a finite automaton. Similar automata are used in grey images and colour images. [10]

Automata and Language has indeed lots of based-application. The study of this field especially its heart – Regular Language is a great concern. There is a great need in having software tools in education according to Mirotznik. [12] In fact, there are lots of applications that help and aid in understanding theoretical concepts of this field.

In 1998, Charmaine Cristi utilized the object-oriented resource of JAVA in creating Computer-Aided instruction on the conversion of regular expressions to their equivalent Non-deterministic Finite Automata with ε-transitions and Non-deterministic Finite without ε-transitions. According to the author, it is very tedious to do the said tasks manually so she created a system that could aid and help in understanding the concepts involved. There three parts that the system used, namely, the analyzer, the converter, and the graphics generator. The analyzer covers the parsing of the input and groups them into tokens. The converter aided the equivalent NFA of the inputted RE. The graphics generator displays the transitions diagram on the display screen as well as its equivalent machine. [13]

In 2002, another Computer-Aided instruction System on the automation and simulation for automata with outputs was developed by Marvin Gimenez. According to Gimenez, Automata and Language theory involves tedious computations that are prone to error if done manually and creating Automata with outputs system will aid better understanding of these concepts. The system consists of these tasks: creation of a Moore/Mealy machine, conversion of a Moore machine into its equivalent Mealy machine and vice versa, and conversion of an input string into its corresponding output string for a defined machine. Also, the system consists of three main parts like Cristi's, the parser, converter, and graphic generator. The author also used JAVA in creating the system. [14]

III. THEORETICAL FRAMEWORK

A. Regular Language

Regular Languages are exactly the ones that can be described by finite automata [1] .It is a formal language that satisfies the property of being accepted by Finite Automaton, described by a Regular Expression, generated by Regular Grammar and Prefix Grammar, accepted by read-only Turing Machine and defined by monadic second order logic . These notations, that describe a regular language, are known to be equivalent in nature.

The collection of regular languages over an alphabet Σ is defined recursively as follows:

- the empty language Ø is a regular language.
- the empty string language { ε } is a regular language.
- For each *a* ϵ Σ, the singleton language { *a* } is a regular language.
- If *A* and *B* are regular languages, then *A* Ụ *B* (union), *A* • *B* (concatenation), and *A** (Kleene star) are regular languages.
- No other languages over Σ are regular. [15]

The typical representations of a Regular Language are the Finite Automata and the Regular Expressions.

1. Finite Automaton

A finite automaton has a set of states, and its "control" moves from state to state in response in external inputs. [1] The crucial part of determining the distinctions of classes of the finite automaton is whether it is "deterministic" meaning it can only be at one state at a time or “non-deterministic” meaning it can be in more than one state at a time. We will see that imposing non-determinism to a finite automaton can also be described by a deterministic finite automaton but there can be considerable efficiency in describing an application using a non-deterministic automaton. A finite automaton consists of the following:

1. A finite set of states, often denoted by Q.
2. A finite set of *input symbols*, often denoted by Σ
3. A *transition function* that takes as argument a state and an input symbol and returns a state or group of states. The transition function will commonly be denoted δ. It is represented by arcs between states and the labels on the arcs. If q is a state, and an a is an input symbol, then δ(q,a) is that state p such that there is an arc labeled from q to p.
4. A start state, one of the states in Q.
5. A set of final or accepting states in F. The set F is a subset of Q.

Thus a finite automaton can be represented by the five-tuple notation:

$$A = (Q, \Sigma, \delta, q0, F)$$

Specifying a finite automaton using the five-tuple notations with complete definition of the transition functions is very tedious and hard to read and there are two preferred notations for describing finite automata:

1. A *transition diagram,* which is a graph with states represented by a circle and transition function represented by an arc.
2. A transition table, which is a tabular listing of the δ function, which by implication tells us the set of states and the input alphabet. [1]

a. **Deterministic Finite Automaton**

Deterministic refers to an instance wherein on each input there is one and only one state which the automaton can shift from its current state. A transition diagram for a DFA A = (Q, Σ, δ, q0, F) is a graph defined as follows:

a) For each state in Q there is a node.
b) For each state q in Q and each symbol a in Σ, let δ(q, a) = p. Then the transition diagram has an arc from node q to node p, labeled a. If there are several input symbols that cause a transition from q to p then the transition diagram can have one arc, labeled by the list of these symbols.
c) There is an arrow into the start state q0, labeled *Start*. This arrow does not originate at any node.
d) Nodes corresponding to accepting states (those in F) are marked by double circle. States not in F have single circle. [1]

DFA can also be represented by a transition table which is usual, tabular representation of a function like δ that takes two arguments and returns a value. The rows of the table represent the states, and the columns represent the inputs. The entry for the row corresponding to state q and the column corresponding to input *a* is the state δ (q, a).

A DFA defines a language that is the set of all strings that result in a sequence of transition goes from the start state to any accepting states. Making the notion of a DFA precise, an extended transition function is implemented. It describes what happens when we start in any state and follow any sequence of inputs. If a transition function is denoted by δ then the extended transition function is denoted by $\delta^{\wedge}$ (delta-hat).

b. **Non-deterministic Finite Automaton**

A "non-deterministic" finite automaton (NFA) has the power to be in several states at once. This ability is often expressed as the ability to "guess" something about its input. [1] The only difference of an NFA to a DFA is its extended transition function. As for the DFA's we need to extend the transition function δ of an NFA to a function $\delta^{\wedge}$ that takes a state *q* and a string of input symbols *w,* and returns the set of states that the NFA is in if it is starts in state *q* and processes the string *w*. Thus, it is somewhat taking the union of the states from the transition given an input *a* to state q. Meaning, the language accepted by an NFA is a string which has possibility to make any sequence of choices of next state, while reading the characters, and go from the start state to at any accepting state. The fact that the other choices using the input symbols lead to a non accepting state, or die (do not lead to any state at all), does not prevent the string to be accepted by the NFA as a whole [1].

c. **Non-deterministic Finite Automaton with ε-transitions**

Another extension of the finite automaton is a new feature that allows a transition on ε, the empty string. In effect, an NFA is allowed to make a transition spontaneously without receiving an input symbol. Like the non-determinism added, this new capability does not expand the class of language that can be accepted by finite automata, but it gives some added "programming convenience." We shall also see later how NFA's with epsilon transition which we call ε-NFA's, are closely related to regular expressions and useful in proving the equivalence between the classes of languages accepted by finite automata and by regular expressions. [1] We can represent ε – NFA exactly as NFA except for one, the transition function must include information about transition on ε. We represent ε-NFA by $A = (Q, \Sigma, \delta, q_0, F)$ where all components are the same as NFA, but δ is now a function that takes as argument:

1. A state in Q, and
2. A member of Σ U {ε}, that is either an input symbol, or the symbol ε. We may require that ε, the empty string, cannot be a member of the alphabet Σ, so no confusion results.[1]

We need to learn a substantial definition called the ε-closure of a state in computing ε-NFA's. Informally, ε-closure ECLOSE(q) composed of the state itself q and recursively all other states from q that has an ε transition.

2 . Regular Expression

Another notation that can describe a regular language is the algebraic description called "regular expression". We shall see that the regular expression defines exactly the same language as the various forms of automata describe. However, regular expressions offer something that is not visible in automata, the declarative way to express the strings we want to accept. It somewhat serve as the input language for many systems that process a string. Regular Expressions denotes a language using some operators:

1. The *union* of two languages L and M, denoted L Ụ M, is the set of strings that are in either L or M, or both.
2. The *concatenation* of Language L and M is the set of strings that can be formed by taking any string in L and concatenating it with any string in M. We denote concatenation with a dot or no operator at all.

3. The *closure(or star or Kleene closure)* of a language L is denoted L * and represents the set of those strings that can be formed by taking any number of strings from L, possibly with repetitions and concatenating all of them.

The language denoted by regular expressions consists of three basis:

1. The constant ε and ø are regular expressions.
2. If a is any symbol, then a is a regular expression.
3. A variable, usually capitalized and italic such as L, is a variable, representing any language.

Some algebraic laws are involved in simplifying or expressing language formed by regular expressions some of which are:

a. L + M = M + L. This law, the commutative law for union, says that we may take the union of two languages in either order.
b. (L + M) + N = L + (M + N). This law, the associative law for union, says that we may take the union of three languages either by taking the union of the first two initially, or taking the last two initially.
c. (LM)N = L(MN). This law, the associative law for concatenation, says that we can concatenate three languages by concatenating either the first two or the last two initially.
d. Ø + L = L + ø = L. This law asserts that ø is the identity for union.
e. εL = Lε = L. This law asserts that ε is the identity for concatenation.
f. øL = øL = ø. This law asserts that ø is the annihilator for concatenation.
g. L(M + N) = LM + LN. This law is the left distributive law of concatenation over union.

h. $(M + N) L = ML + NL$. This law, is the right distributive law of concatenation.

i. $L + L = L$. This law, is the idempotence law for union, states that if we take the union of two identical expressions, we can replace them by one copy of the expression.

j. $(L^*)^* = L^*$. This law says that closing an expression that is already closed does not changed the language.

k. $Ø^* = ε$. The closure contains only the string ε.

l. $ε^* = ε$. It is easy to check that the only string formed by concatenating any number of copies of the empty string is the empty string tself.

m. $L^+ = LL^* = L^*L$. L^+ is defined to be $L + LL + LLL +\dots$ Also $L^* = ε + L + LL + LLL +\dots$ Thus, $LL^* = Lε + LL + LLL + LLLL+ \dots$

n. $L^* = L^+ + ε$.

o. $L? = ε + L$. This is definition of the ? operator[1].

3. Conversion

a. From NFA to DFA

The language accepted by a DFA is exactly the same language accepted by an NFA. Thus, we can conclude or it is not surprising that there is equivalence of DFA and NFA. The conversion involves an important “construction” called the *subset construction* because it involves constructing all subsets of the set of states of the NFA.

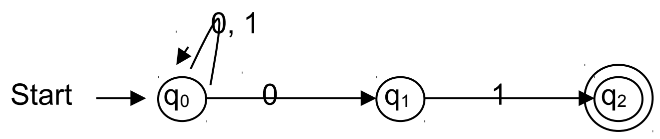
The subset construction starts from an NFA $N = (Q_N, Σ, δ_N, q_0, F_N)$. Its goal is the description of a DFA $D = (Q_D, Σ, δ_D, \{q_0\}, F_D)$ such that $L(D) = L(N)$. Notice that the input alphabets of the two automata are the same and the start state of D is the set containing only the start state of N. The other components of D are constructed as follows.

- Q_D is the set of subsets of Q_N; i.e., Q_D is the *power set* of Q_N. Note that if Q_N has n states, then Q_D will have 2^n states. Often, not at all these states are accessible from the start states, then Q_D will have 2^n states. Often, not all these states are accessible from the start state of Q_D. Inaccessible states can be "thrown away," so effectively, the number of states of D may be much smaller than 2^n.
- F_D is the set of subsets S of Q_N such that $S \cap F_N \neq ø$. That is, F_D is all sets of N's states that include at least one accepting state of N.
- For each set S subset of Q_N and for each input symbol *a* in Σ.

$$\delta_D(S, a) \;=\; \bigcup_{p \text{ in } S} \delta_N(p,a)$$

That is to compute δ_D (S, a) we look at all the states p in S, see what states N goes to from p on input , and take the union of all those states.[1]

The subset construction is very tedious to do knowing that not all the states in the subset are relevant or accessible from the start state of the automaton. We may use "lazy evaluation" to decrease the burden of performing transitions of each state in the subset. Lazy Evaluation performs only the transition in the state that comes up performing the previous transition function from the start state.



i.e. An NFA that accepts all string ending in 01.

Transition table:

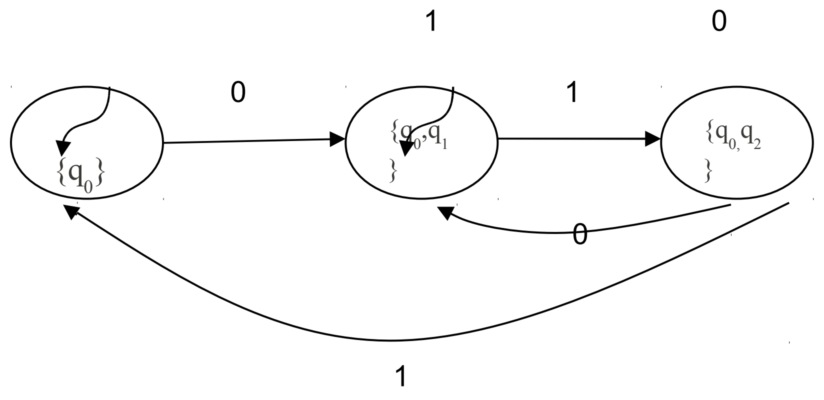
Subset Construction:

	0	1
ø	ø	ø
→ {q_0}	{q_0, q_1}	{q_0}
{q_1}	ø	{q_2}
*{q_2}	ø	ø
{q_0, q_1}	{q_0, q_1}	{q_0, q_2}
*{q_0, q_2}	{q_0, q_1}	{q_0}
*{q_1, q_2}	ø	{q_2}
*{q_0, q_1, q_2}	{q_0, q_1}	{q_0, q_2}

By Lazy Evaluation:

	0	1
→ {q_0}	{q_0, q_1}	{q_0}
{q_0, q_1}	{q_0, q_1}	{q_0, q_2}
*{q_0, q_2}	{q_0, q_1}	{q_0}

The transition diagram of the DFA equivalent to the given NFA is:



b. From ε-NFA to DFA

Suppose that E = (Q, Σ, δ, q_0, F) is an ε-NFA. We first define $\delta^{\wedge}$, the extended transition function, to reflect what happens on a sequence of inputs. The intent is that $\delta^{\wedge}$(q,w) is the set of states that can be reached along *a* oath whose labels, when concatenated, form the string *w*. The appropriate recursive definition is $\delta^{\wedge}$(q, ε) = ECLOSE(q). That is, if the label of the path is ε, then we can follow only ε-labeled arcs extending from state q; that is exactly what ECLOSE does. Given any ε-NFA E, we can find a DFA D that accepts the same language as E. The construction we use is very similar to the subset construction, as the states of the D are subsets of the states of E. The only difference is that we must incorporate ε-transitions of E, which we do through the mechanism of the ε-closure.[1]

Consider the following ε-NFA.

	ε	a	b	C
p	ø	{p}	{q}	{r}
q	{p}	{q}	{r}	Ø
*r	{q}	{r}	ø	{p}

Taking the E-Closure of each state:

ECLOSE (p) = {p}

ECLOSE (q) = {p, q}

ECLOSE (r) = {p, q, r}

By (1) using Lazy Evaluation in Subset Construction and,

(2) taking the E-CLOSURE of the result in (1) to eliminate ε:

Starting from the start state:

Start state of DFA = ECLOSE(p) {p →start state of ε-NFA} = **{p}**

State **{p}**

input a: δ(p, a) = {p} ECLOSE(p) = **{p}**

input b: δ(p, b) = {q} ECLOSE(q) = **{p,q}** [new state] input c: δ(p, c) = {r} ECLOSE(r) = **{p, q, r}** [new state]

State **{p, q}**

input a: δ(p, a) U δ(q, a} = {p, q}

ECLOSE(p) U ECLOSE(q) = {p} U {p, q} = **{p, q}**

input b: δ(p, b) U δ(q, b} = {q, r}

ECLOSE(q) U ECLOSE(r) = {p, q} U {p, q, r} = **{p, q, r}**

input c: δ(p, c) U δ(q, c} = {r}

ECLOSE(r) = **{p, q, r}**

State **{p, q, r}**

input a: δ(p, a) U δ(q, a} U δ(r, a) = {p, q, r}

ECLOSE(p) U ECLOSE(q) U ECLOSE(r)

= {p} U {p, q} U {p, q, r} = **{p, q, r}**

input b: δ(p, b) U δ(q, b} U δ(r, b) = {q, r}

ECLOSE(q) U ECLOSE(r)

= {p, q} U {p, q, r} = **{p, q, r}**

input c: δ(p, c) U δ(q, c} U δ(r, c) = {p, r}

ECLOSE(p) U ECLOSE(r)

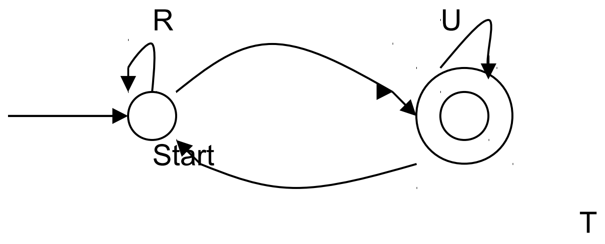
= {p} U {p, q, r} = **{p, q, r}**

The transition table of a DFA equivalent to the given ε-NFA is

	a	b	c
{p}	{p}	{p, q}	{p, q, r}
{p, q}	{p, q}	{p, q, r}	{p, q, r}
*{p, q, r}	{p, q, r}	{p, q, r}	{p, q, r}

c. **From DFA to RE**

It is known that finite automata and regular expressions accept the same language – regular language. Converting from a DFA to regular expression involves some processes. One of the methods used to convert is the state elimination technique using one of the expressions (R + SU*T)*SU* denoted by a generic two-state automaton:



(2) In explanation we can go from the start state to itself any number of times, by following a sequence of paths whose labels are in either L(R) or L(SU*T). The expression SU*T represents paths that go to the accepting state via a path in L(S), perhaps return to the accepting state several times using a sequence of paths with labels in L(U), then return to the start state with a path whose label is in L(T). Then we must go to the accepting state, never to return in the start state, we can return to it as many times as we like, by following a path whose label is in L(U).

The expression R + QS*P is the result of eliminating a state and connecting the predecessor to the successor of the state to be eliminated where:

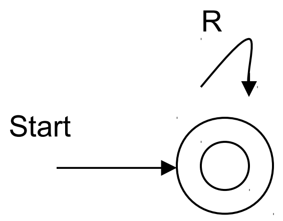
R is the direct route from the predecessor to the successor,

Q is the arc from the predecessor to the state,

S is the loop in the state to be eliminated,

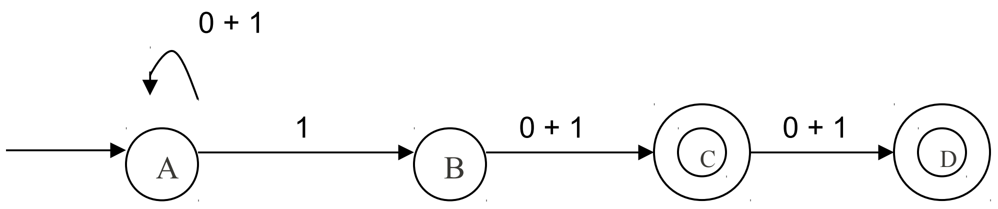
P is the arc from the state to the successor

(3) If the start state is also an accepting state, then we must also perform a state elimination from the original automaton that gets rid of every state but the start state. When we do so, we are left with a one-state automaton that looks like this:



Finally, the desired output is the sum (union) of all the expressions derived from the reduced automata for each accepting state, by rules (2) and (3).

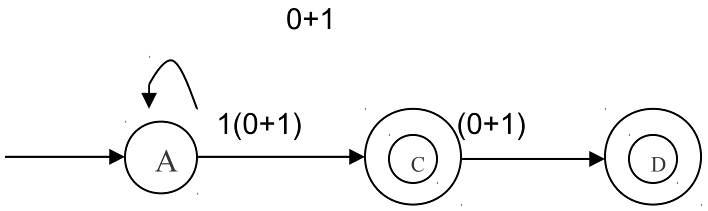
Consider the NFA accepting strings that have a 1 either two or three positions from the end:



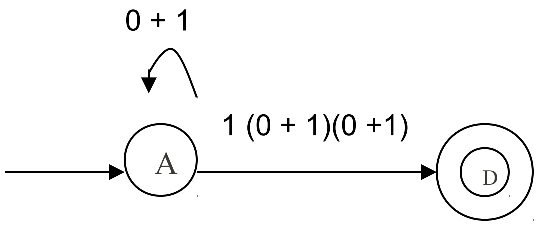
Eliminating B since it is neither a start state nor an accepting state, R = Ө (since the arc from A to C does not exist), Q = 1 (arc from A to B), S = Ө (since there is no loop in B), P

= (0 + 1) (arc from B to C). The expression resulting from eliminating B is **ϴ + 1ϴ*(0+1).** To simplify we may use the laws mentioned in the Regular Expression. Since union of null to an expression is the expression itself (law **d**) and multiplying L(ϴ*) = (ε) U L(ϴ) U L(ϴ) … = ε to an expression is the expression itself (law **e**), the simplified form is **1(0+1)**..

The resulting graph with B eliminated is:

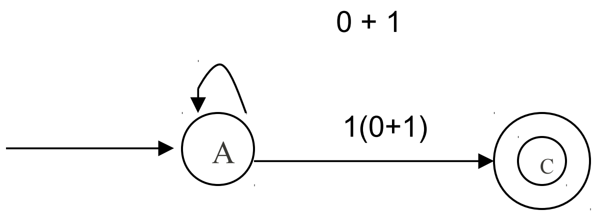


Now we must branch, eliminating states C and D in separate reductions. To eliminate state C, the mechanics are similar to those we performed above to eliminate state B, and the resulting automaton is:



In terms of the generic two state automaton of the Regular Expressions are: R = 0 +1, S = 1(0+1)(0+1), T = θ, U = θ. The expression U* can be replaced by ε, i. e., eliminated in a concatenation is the justification is that θ* = ε. Also, the expression SU*T is equivalent to θ since T, one of the terms of the concatenation is θ. The generic expression (R + SU*T)*SU* thus simplifies in this case to R*S, or **(0+1)*1(0+1)(0+1).**

Now let us eliminate B from the previous graph. The resulting graph will now be:



We can apply the rule for two-state automata to simplify the expression to get **(0 + 1)*1(0 + 1).**

All that remains is to sum the two expressions to get the expression for the entire automaton. This expression is

$$\mathbf{(0 + 1)^*1(0 + 1) + (0+1)^*1(0+1)(0+1)}$$

Note: we may also use the expression below to convert a DFA to Regular Expression

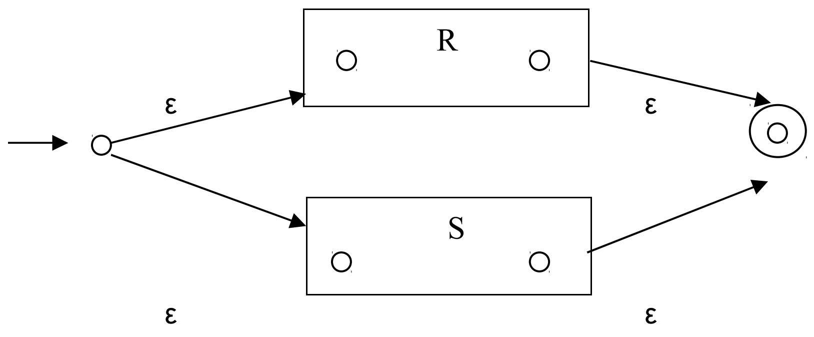
$$R_{ij}^{(k)} = R_{ij}^{(k-1)} + R_{ik}^{(k-1)}(R_{kk}^{(k-1)})^*R_{kj}^{(k-1)}$$

for the labels of all paths from state I to j that go through no state higher than k. If we construct these expressions in order of increasing superscript, then since each $R_{ij}^{(k)}$ depends on expressions with a smaller superscript, then all expressions are available when we need them. Eventually we have $R_{ij}^{(n)}$ for all i and j (n =num of states). The regular expression for the language is the sum of all expressions $R_{1j}^{(n)}$ such that 1 is the start state and j is an accepting state.[1]

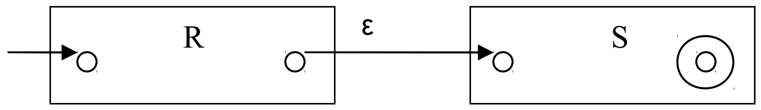
d. **From RE to ε-NFA**

On the other hand, we must understand the way of converting a regular expression to an automaton. The easiest way is to convert the regular expression to an ε-NFA. This summarizes the possible simplest arithmetic on a regular expression:

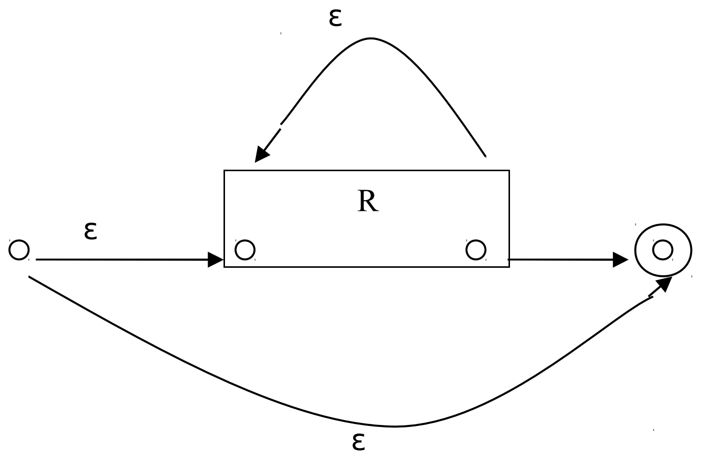
1. R + S



2. RS



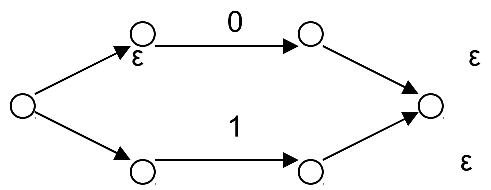
3. R*



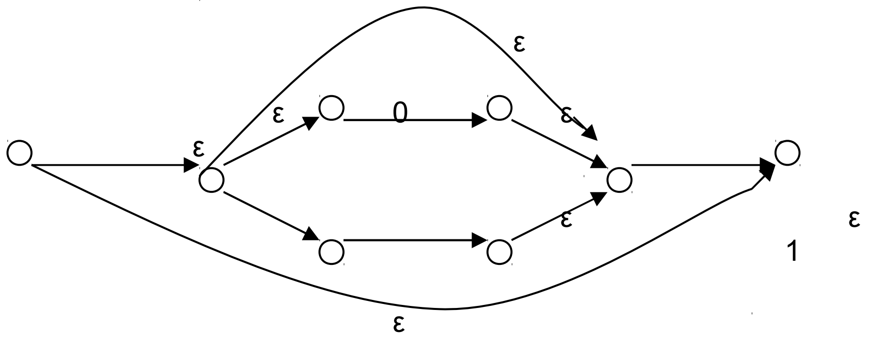
Consider the regular expression **(0 + 1)*1(0+1)**.

We can convert it to ε – NFA by constructing the automaton of each terms and connecting them. The resulting automaton following the construction above (1 – 3) is:

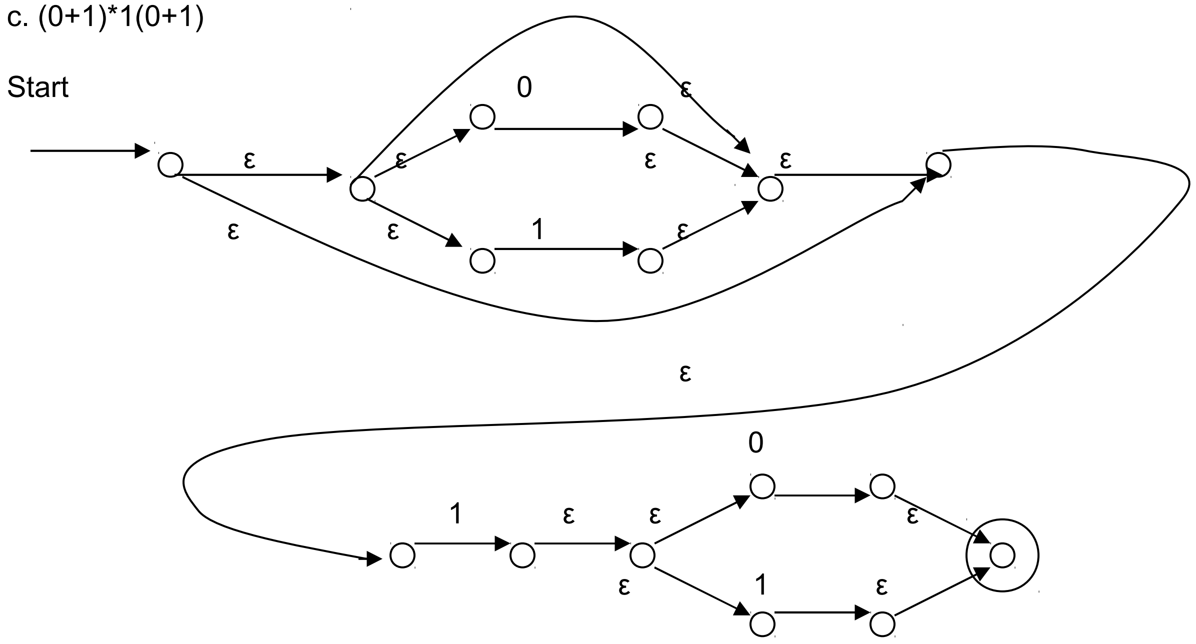
a. (0 +1)



b. (0+1)*



c. (0+1)*1(0+1)



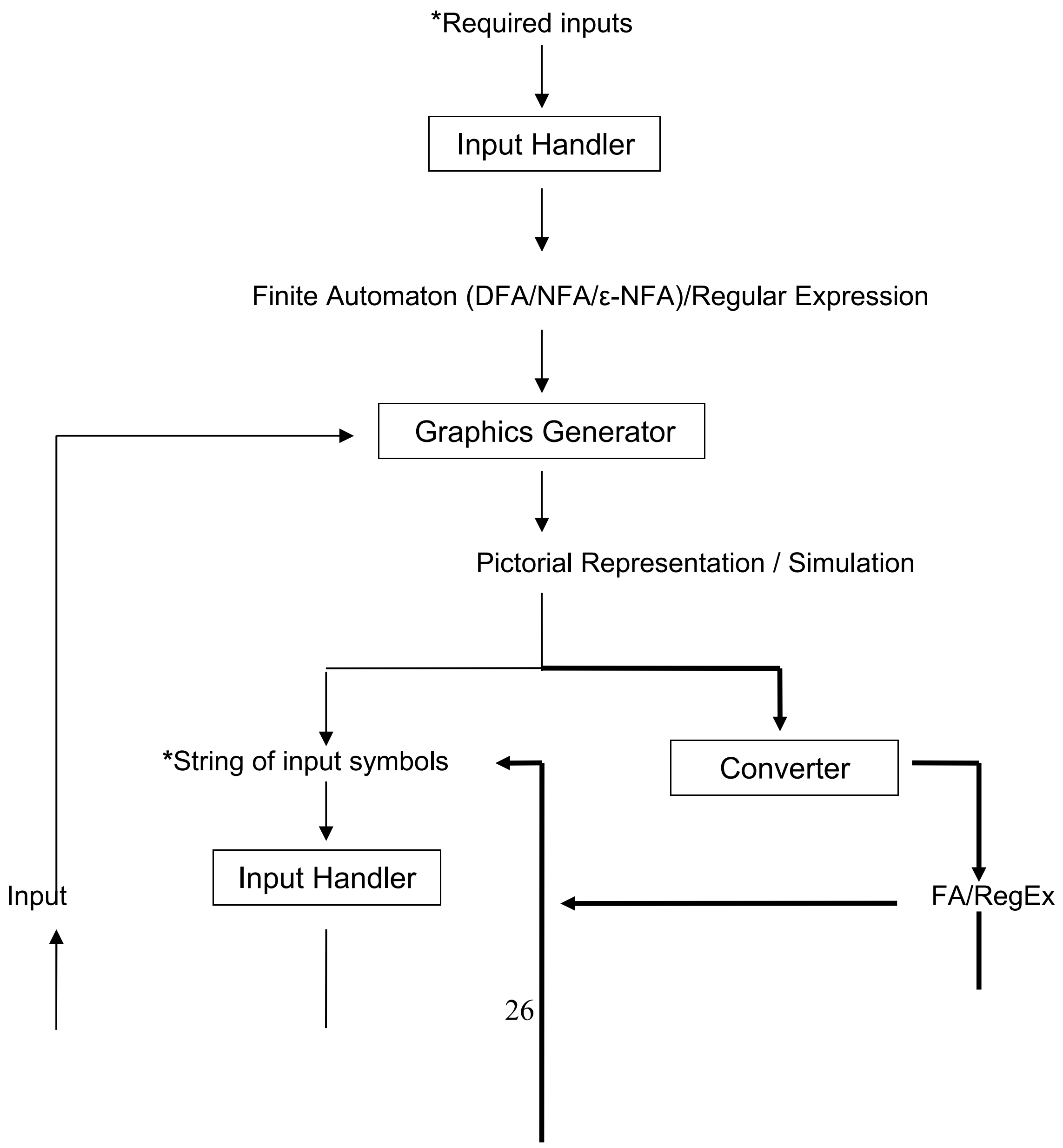
The ε-NFA above(c) is the exact equivalent of **(0+1)*1(0+1)**.

B. E-LEARNING

E-Learning has become a major field of interest in recent years, and multiple approaches and solutions have been developed. A typical form of e-learning application comprises exercise submission and assessment systems that allow students to work on assignments whenever and where they want (i.e., dislocated, asynchronous work). [16]

IV. DESIGN AND IMPLEMETATION

The REND System has three important parts: the input handler, the graphics generator, and the converter (See Figure 1).



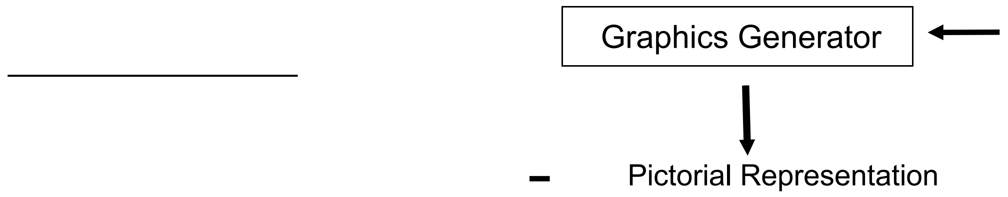


Figure 1: A diagram that summarizes Automation and Simulation with Conversion of REND System

*	represents user's input
▬	Bold lines mean the process can only be done once

The REND's context diagram and top data flow diagram are shown in Figures 2 & 3 respectively. The Context Diagram shows that the User must supply the required data for the REND System to utilize. Consequently, the REND System performs computations and conveys the output to the User. REND has five main processes as shown in the Top Data Flow Diagram namely Getting and Checking of Inputs, Creating an Automaton, Conversion, Generating Graphics and the Simulation of an Input String. The inputs are in the form of a dropdown table which serves as the transition table. After completing the table, the REND System will automatically create an automaton for the user to view. The user has a choice of converting an automaton or tests an input string if it is part of the language. After the conversion processes, the user may view the graphic result of conversion.

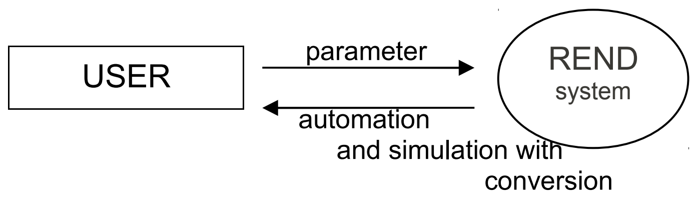
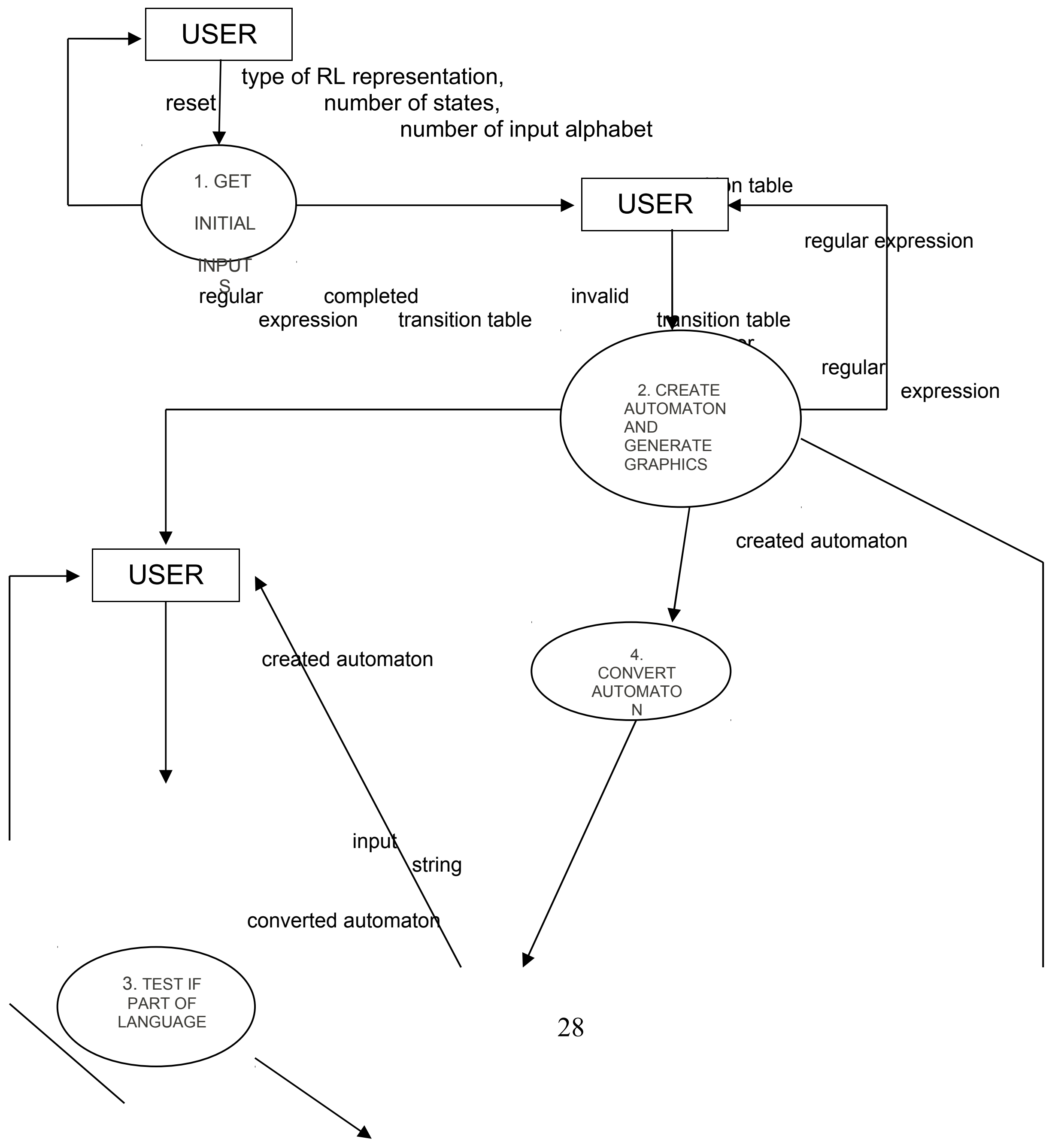


Figure 2: Context Diagram of REND


USER
type of RL representation,
reset
number of states,
number of input alphabet
1. GET INITIAL INPUTS
USER
on table
regular expression
regular
expression
completed
transition table
invalid
transition table
regular
expression
2. CREATE AUTOMATON AND GENERATE GRAPHICS
created automaton
USER
created automaton
4. CONVERT AUTOMATON
input
string
converted automaton
3. TEST IF PART OF LANGUAGE

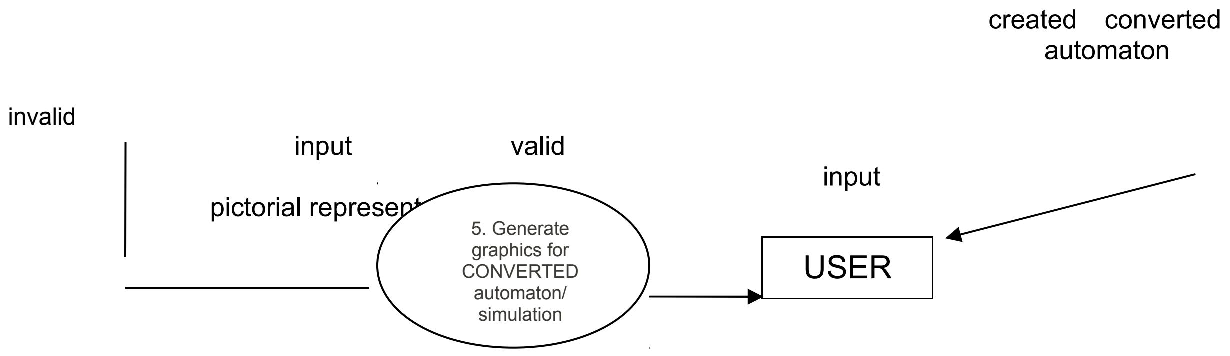

created converted automaton
invalid
input
valid
input
pictorial represent
5. Generate graphics for CONVERTED automaton/ simulation
USER

Figure 3: Top Data Flow Diagram of REND

A. **Input Handler**

The input handler executes the reading and analyzing of input. It is first implemented by generating a graphical interface with a radio button for the kind of representation (DFA, NFA, ε-NFA, Regular Expression, a combobox for the number of states. After getting the required information, new form appears above the initially generated components. If the user choose a finite automaton (DFA, NFA, ε-NFA), a table consisting of comboboxes becomes visible with states and input symbols being the row and column respectively. Otherwise four dropdown comboboxes appears to handle the regular expression.

Checking of input is also implemented. It checks the number of occurrences of an input symbol from the array of input symbols, a symbol may occur only once. In the simulation process, it checks if the input string contains an invalid symbol or exceeds the maximum allowable length.

B. **Graphics Generator**

After the Input Handler processed the input string, a module called the Graphics Generator provides the automaton's pictorial representation, conversion and the simulation of the automaton's transition events given an input.

A state is represented by a circle; the starting state has a different color(reddish). Every state has a label for its name and transitions are represented by a direct line; above or below it is the input symbol depending on the type of representation. In the simulation part, the user can't see, by animation, the traversal of each state depending on the input string that was provided by the user but it is checked if it is a member or not. Also, a transition table, and log results (both in the side) and a thorough explanation (below) is available for the user to view.

C. **Converter**

The converter handles the conversion of the four possible conversion processes – NFA to DFA, ε-NFA to DFA, DFA to Regular Expression and Regular Expression to ε-NFA.

This module use different procedure in converting one form to another. In the conversion process, the transition table also changes as well as the results in the transition function results and explanation.

1. **From NFA to DFA**

The conversion from a non-deterministic finite automaton to a deterministic finite automaton is done simply by following the subset construction specifically the lazy evaluation process. The inputs that came from the transition table entered by the user are used and underwent the lazy evaluation to fill the new table for the construction of a DFA. Mapping of transition table δ will be done. The system will provide new names for the states of DFA in the transition diagram and have labels below to avoid conflict.

2. From ε-NFA to DFA

An array list of a class holds first the ECLOSURE of each state in the NFA with epsilon transition. Converting from non-deterministic finite automaton with ε transition to a deterministic finite automaton also follows the lazy evaluation technique but this time the ECLOSURE is applied as well. Mapping of transition table δ will be done. The new table is used in the generation of the new diagram. Also, new names are assigned to the transition diagram.

3. From DFA to Regular Expression

The conversion is done by elimination method. Following this method, the predecessor of the state to be eliminated connects to the successor of the state and the label for this route is now the concatenation of the input from the predecessor to the state and the input from the state to the successor. This is done recursively until there are only two states left, the start and an accepting state. It is possible to have two accepting states or more. The results are concatenated (+) to get the final output-the generated regular expression.

4. From Regular Expression to ε-NFA

The system defined graph handling the possible terms in the regular expression (concatenation, union and Kleene star) for this kind of conversion will just be utilized. Then the required system defined-graphs are just concatenated by an arrow.

D. Technical Architecture

System requirements (minimum)

512 mb ram

.Net Framework 2.0 (included in installation files)

Microsoft.Directx.Direct3d.dll (included in installation files)

Microsoft.Directx.dll (included in installation files)

V. RESULTS

The main page of the system (shown in Figure 4) allows you to launch REND Visualization Tool (Figure 5), Reference, How to Use, and Assessment. The REND Visualization Tool Frame displays the main functionalities of the system: to create a Finite Automaton to populate the transition table, to convert the created automaton to its equivalent representation, to create a Regular Expression and to convert it to equivalent NFA.

To create an automaton, click File in the menu, click new and select your desired automaton (illustrated on Figure 6). A new frame will appear and enables the user to choose the number of states. After clicking ok, another frame will appear with a table composing of dropdown comboboxes(Figure 7). This will serve as the transition table and the user is required to fill in the table. This table checks if a start state and a final state is specified by the user. An error will be displayed if certain requirement is not met. By clicking View Diagram button, the result of creating an automaton will be displayed in a panel(as shown in Figure 8). After entering an automaton, the user can convert it by clicking Convert, and then select the desired conversion (Figure 9). If conversion is successful, the system will inform the user. By clicking the View Diagram button again, the output will be displayed in the panel(illustrated on Figure 10). The user can enter a regular expression by clicking File, New, then Regular Expression(Figure 11). This will display new form with 4 comboboxes with one textfield. After selecting the desired regular expression of each of the combobox, the textfield will automatically update by concatenating the inputs of the comboboxes. The system auto-converts the regular expression to an automaton(Figure 12). The user has to click the view diagram to see the output. The position of diagram is not static and hence the user can click the view diagram several times until the desired graphical representation layout is met.

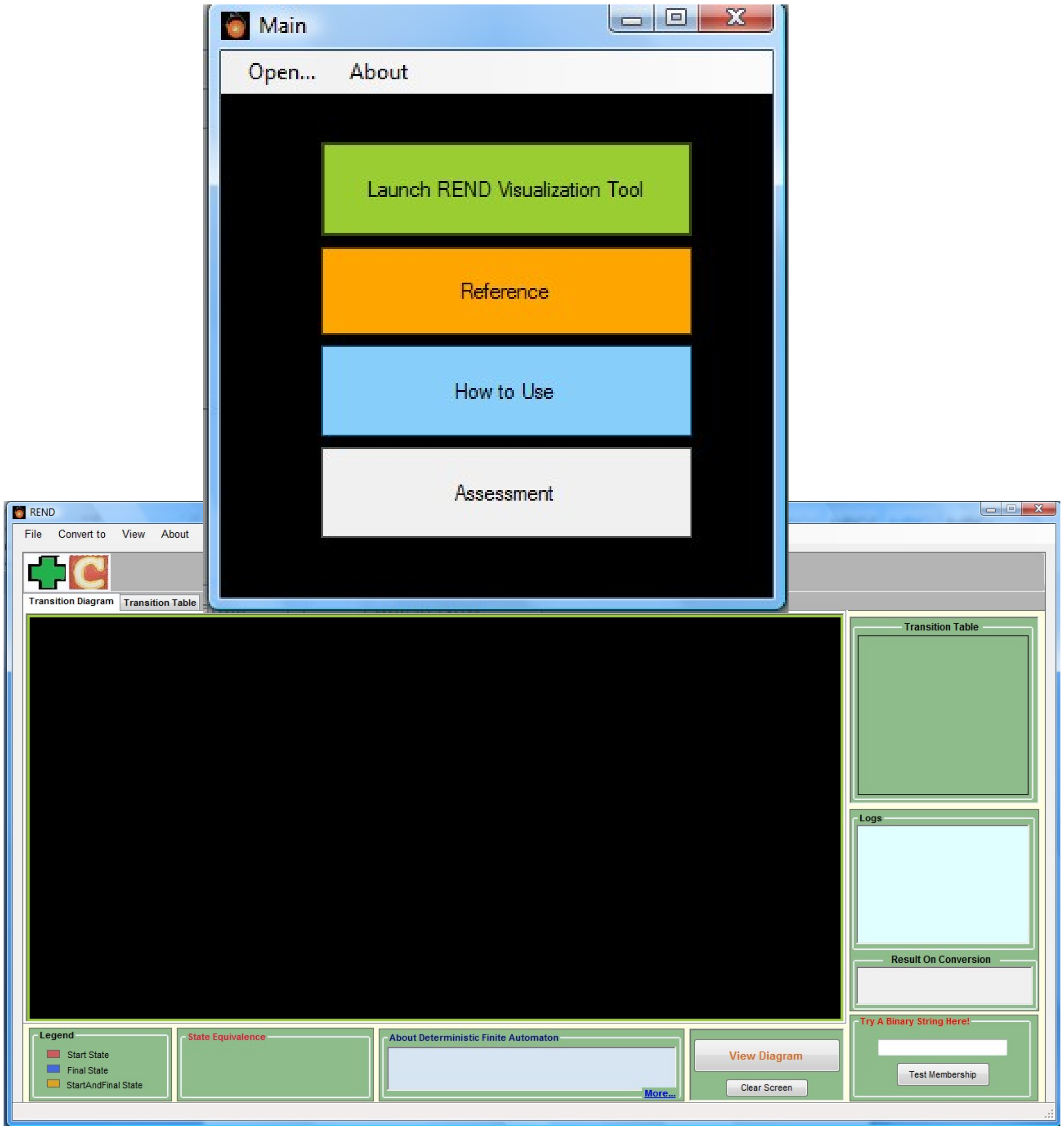


Figure 5: Visualization Frame of REND

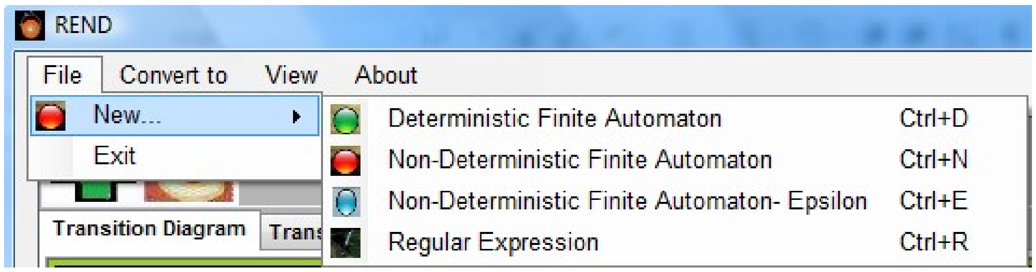
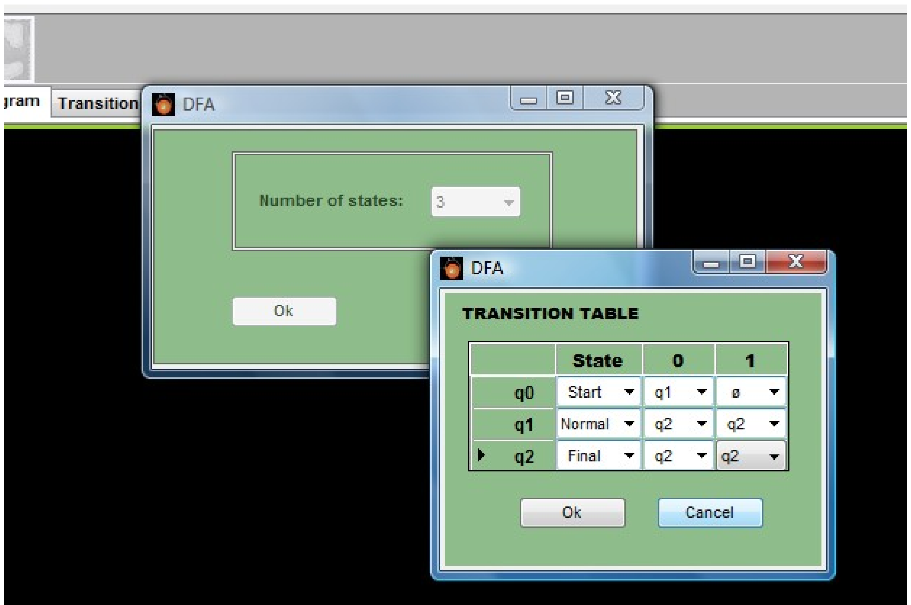

REND
File
Convert to
View
About
New...
Exit
Transition Diagram
Trans
Deterministic Finite Automaton Ctrl+D
Non-Deterministic Finite Automaton Ctrl+N
Non-Deterministic Finite Automaton- Epsilon Ctrl+E
Regular Expression Ctrl+R

Figure 6: Creating an Automaton of REND



Number of states: 3

TRANSITION TABLE

	State	0	1
q0	Start	q1	ø
q1	Normal	q2	q2
q2	Final	q2	q2

Figure 7: Populate the Transition Table of REND

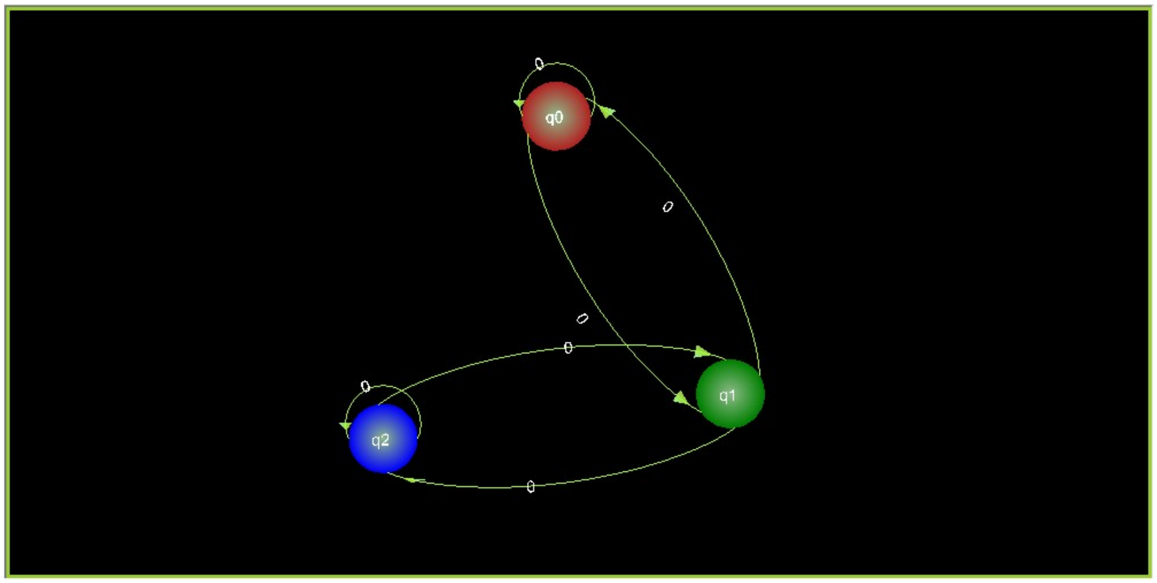


Figure 8: Result of Creating an Automaton of REND

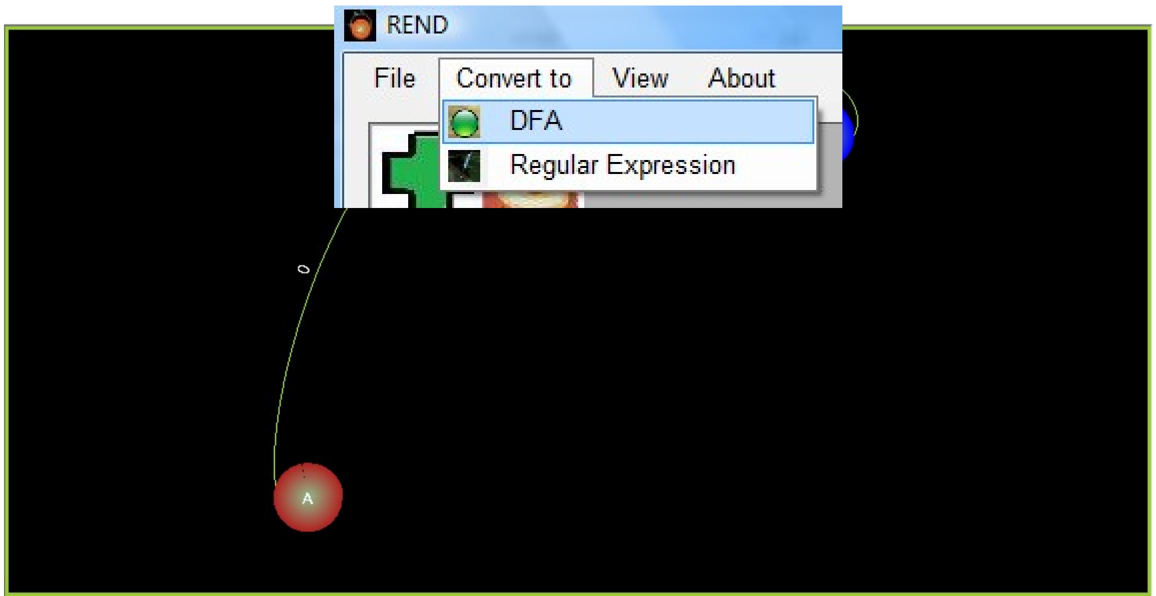
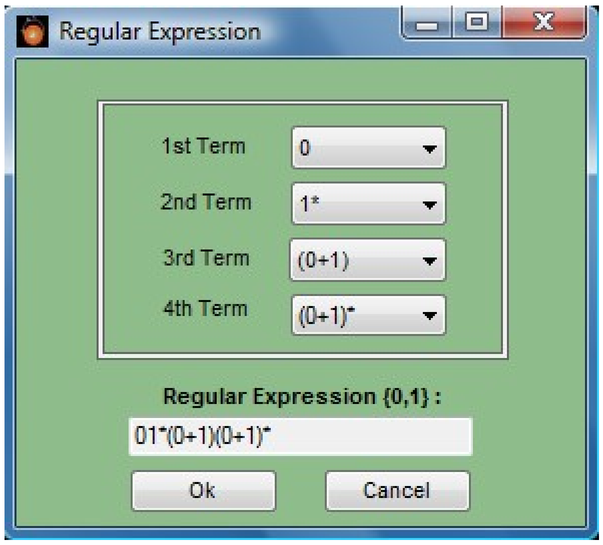


Figure 10: Sample Conversion Result of REND


Regular Expression

1st Term	0
2nd Term	1*
3rd Term	(0+1)
4th Term	(0+1)*

Regular Expression {0,1} :

01*(0+1)(0+1)*

Figure 11: Creating a Regular Expression of REND

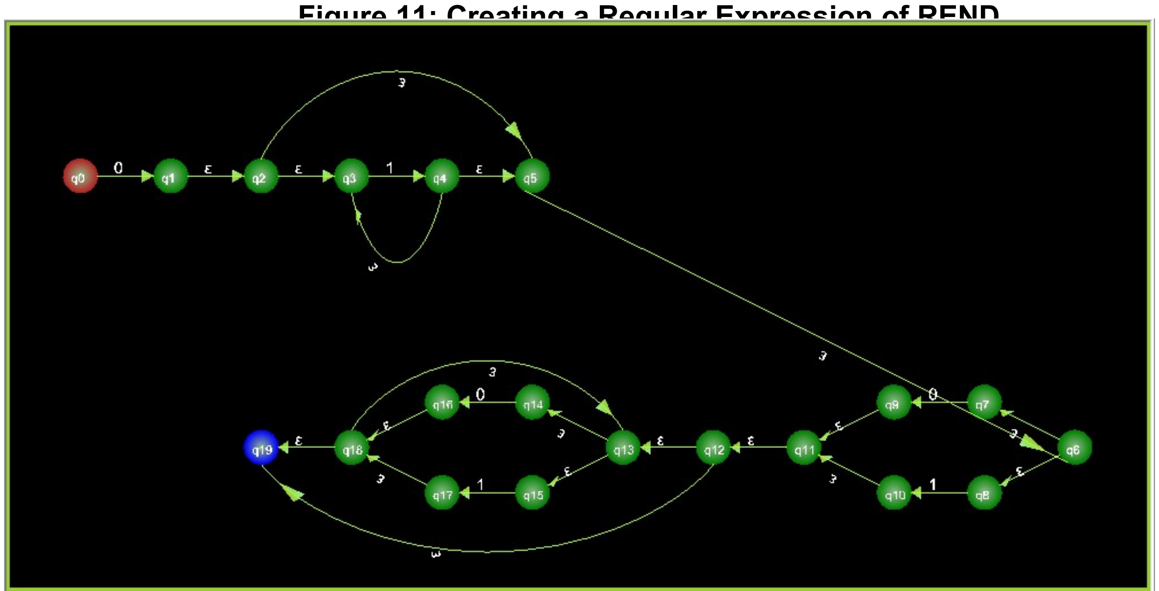


Figure 12: Result of Converting a Regular Expression to Automaton of REND

Input String can also be tested to check if it is a member of the language described by a finite automaton or regular expression (Figure 13). The input is checked automatically if it consists only of binary alphabet, zero and one, if not it will not recognize it and it therefore will not be displayed. If an input string is part of the language then the system will inform the user(Figure 14)

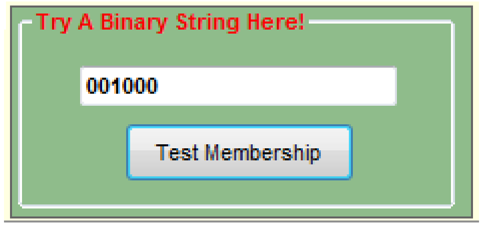
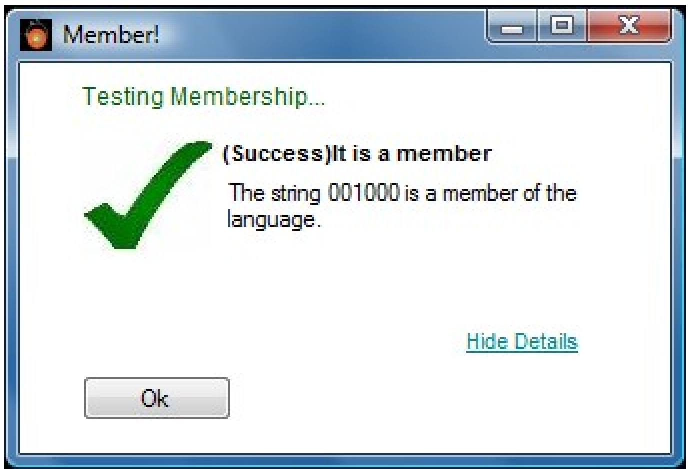


Figure 13: Testing Membership of REND


Member!

Testing Membership...

(Success)It is a member

The string 001000 is a member of the language.

Figure 14: Result of Testing Membership of REND

VI. DISCUSSION

The REND Visualization Tool is system that automates the possible conversion of the four representation of a regular language: DFA, NFA, ε- NFA, and regular expression. The user inputs a finite automaton or a regular expression and then can be converted to the possible conversion. The possible conversions that can be done are: from NFA to DFA, from ε-NFA to DFA, DFA to Regular Expression, and From Regular Expression to ε-NFA. After entering a representation in the system, the user can enter an input string and let the system check if it is a member of the language or not. This system includes tutorial, manual and an assessment exam for the user to utilize.

One advantage of the system is that it gives the user the privilege of not performing the manual computations for the conversion that is considerably tedious. The animation of the visualization also gives a good layout or form because the system utilizes an algorithm called Force Layout. The system also includes an interactive reference that can be used by the user for reviewing lessons about regular language.

There are disadvantages of the system. The system has a certain limit of number of states in finite automaton and also in the number of terms in regular expression to be performed. Also the system can only handle binary alphabet. It can be assumed that the user will not anymore perform manual computation which is important in developing skills. Also, this system uses a lot of parsing.

E-Learning has become a major field of interest in recent years, and multiple approaches and solutions have been developed. Applying this in acquiring knowledge will be of great benefit to the user and their understanding of some theories involving the Automata and Regular Language through simulation of different forms of regular language, detailed tutorial in the said topics and self assessment exams.

VII. CONCLUSION

The REND Visualization Tool enables the user to convert one representation of Regular Language to another. Also, the system simulates an input string to test membership. These features of the system enable the user to save their time and effort for doing manual computations. The reference with an interface enables the user to recall lessons in an interactive way. It is an easy access tutorial for students and also for beginners who wants to explore the Regular Language. The assessment part enables the user to know how deep his knowledge about the basic topic of Automata is: the Regular Language.

VIII. RECOMMENDATION

Although the REND Visualization Tool achieves its objectives, there are parts of the system that can be improved. The user interface can be made more interactive and more pleasant to the user. Animation can be added for catching the attention of students. Extending the number of states that can be used in the system is very well recommended for professional use. And also making it available online will be at best result as a teaching aid for students.

IX. BIBLIOGRAPHY

1. J. Hopcort, R. Motwani, J. Ullman. Introduction to Automata Theory, Languages, and Computation. 2nd ed. Philippines, 2001.
2. B. Bressler, T. Finley, S. Reading, S. Rodger. Turning Automata Theory into a Hands-on-Course. ACM, Texas, USA. 2006.
3. G. Konstantinids. Computing the edit distance of a regular language, Information and Computation, Vol. 205, No. 9(2007), 1307-1316.
4. J. Hromkovic, G. Schnitger. Comparing the size of NFA's with and without ε-transitions, Theoretical Computer Science, Vol. 380, No. 1-2(2007), 100-114.
5. C. Chia-Hsiang, R. Paige. From regular expressions to DFA's using compressed NFA's, Theoretical Computer Science, Vol. 178, No. 1-2(1997), 1-36.
6. D. Wood, H. Yo-Sub. Obtaining shorter regular expressions from finite-state automata, Theoretical Computer Science, Vol. 370, No. 1-3(2007) 110-120.
7. G. Gramlich, G. Schnitger. Minimizing NFA's and Regular Expressions, Journal of Computer and System Sciences, Vol. 73, No. 6(2007), 908-923.
8. M. Silberztein, INTEX: an FST toolbox, Elsevier Science, Theoretical Computer Science, Vol. 231, pp. 33-46.

9. J. Burge, S. Forrest, K. Ingham, et al. Learning DFA representations of HTTP for protecting web applications, Computer Networks, Vol. 51, No. 5(2007), 1239-1255.

10. H. Jurgensen, L. Staiger, H. Yamasaki, et al. Finite automata encoding geometric figures, Theoretical Computer Science, Vol. 381, No. 1-3(2007), 33-34.

11. W. Pugh, P. Stotts. Parallel finite automata for modeling concurrent software systems, Journal of Systems and Software, Vol. 27, No. 1(1994) 27-43.

12. M. Mirotznik, Tool for Schools, IEEE Journal, 1996, 41-45.

13. C. Cristi, RE2NFA: An Automated Conversion of Regular Expression to Non-Deterministic Finite Automata (S. P.), Philippines: College of Arts and Sciences, University of the Philippines Manila, 1998.

14. M. Gimenez: M2M: An Automation and Simulation System for Automata with Outputs (S. P.), Philippines: College of Arts and Sciences, University of the Philippines Manila, 2002.

15. http://en.wikipedia.org/wiki/Regular_language

16. J. Schwieren, G. Vossen, P. Westerkamp. Using Software Testing Techniques for Efficient Handling of Programming Exercises in an E-Learning Platform, Electronic Journal of E-Learning, Vol. 4, No. 1(2006), 87-94.

17. http://looselycoupled.com/glossary/RDBMS

18. E. Albacea, Algorithm and Analysis Design: Computer Science Lecture Note Series, Philippines: Institute of Computer Science CAS, University of the Philippines Los Baňos, 1994.

X. APPENDIX (Source Code)

Frm_main.designer.vb

```
<Global.Microsoft.VisualBasic.CompilerSer
vices.DesignerGenerated()> _
Partial Class frm_Main
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up
the component list.
    <System.Diagnostics.DebuggerNonUserCo
de()> _
    Protected Overrides Sub Dispose(ByVal
disposing As Boolean)
        Try
            If disposing AndAlso
components IsNot Nothing Then
        components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form
Designer
    Private components As
System.ComponentModel.IContainer

    'NOTE: The following procedure is
required by the Windows Form Designer
    'It can be modified using the Windows
Form Designer.
    'Do not modify it using the code
editor.
    <System.Diagnostics.DebuggerStepThrou
gh()> _
    Private Sub InitializeComponent()
        Dim resources As
System.ComponentModel.ComponentResourceMa
nager = New
System.ComponentModel.ComponentResourceMa
nager(GetType(frm_Main))
        Me.Button2 = New
System.Windows.Forms.Button
        Me.Button3 = New
System.Windows.Forms.Button
        Me.Button4 = New
System.Windows.Forms.Button
        Me.MenuStrip1 = New
System.Windows.Forms.MenuStrip
        Me.OpenToolStripMenuItem = New
System.Windows.Forms.ToolStripMenuItem
        Me.RENDVisualizationToolToolStrip
MenuItem = New
System.Windows.Forms.ToolStripMenuItem
        Me.ReferenceToolStripMenuItem =
New
System.Windows.Forms.ToolStripMenuItem
        Me.HowToUseToolStripMenuItem =
New
System.Windows.Forms.ToolStripMenuItem
        Me.AssessmentToolStripMenuItem =
New
System.Windows.Forms.ToolStripMenuItem
        Me.AboutToolStripMenuItem = New
System.Windows.Forms.ToolStripMenuItem
        Me.AboutRENDToolStripMenuItem =
New
System.Windows.Forms.ToolStripMenuItem
        Me.Button1 = New
System.Windows.Forms.Button
        Me.SaveFileDialog1 = New
System.Windows.Forms.SaveFileDialog
        Me.MenuStrip1.SuspendLayout()
        Me.SuspendLayout()
        '
        'Button2
        '
        Me.Button2.BackColor =
System.Drawing.Color.Orange
        Me.Button2.FlatStyle =
System.Windows.Forms.FlatStyle.Popup
        Me.Button2.Location = New
System.Drawing.Point(56, 110)
        Me.Button2.Name = "Button2"
        Me.Button2.Size = New
System.Drawing.Size(208, 49)
        Me.Button2.TabIndex = 4
        Me.Button2.Text = "Reference"
        Me.Button2.UseVisualStyleBackColo
r = False
        '
        'Button3
        '
        Me.Button3.BackColor =
System.Drawing.SystemColors.ButtonFace
        Me.Button3.FlatStyle =
System.Windows.Forms.FlatStyle.Popup
        Me.Button3.Location = New
System.Drawing.Point(56, 222)
        Me.Button3.Name = "Button3"
        Me.Button3.Size = New
System.Drawing.Size(208, 51)
        Me.Button3.TabIndex = 5
        Me.Button3.Text = "Assessment"
        Me.Button3.UseVisualStyleBackColo
r = False
        '
        'Button4
        '
        Me.Button4.BackColor =
System.Drawing.Color.LightSkyBlue
        Me.Button4.FlatStyle =
System.Windows.Forms.FlatStyle.Popup
```

```
        Me.Button4.Location = New
System.Drawing.Point(56, 165)
          Me.Button4.Name = "Button4"
         Me.Button4.Size = New
 System.Drawing.Size(208, 51)
       Me.Button4.TabIndex = 6
   Me.Button4.Text = "How to Use"
        Me.Button4.UseVisualStyleBackColo
r = False
        '
        'MenuStrip1
        '
        Me.MenuStrip1.Items.AddRange(New
System.Windows.Forms.ToolStripItem()
{Me.OpenToolStripMenuItem,
Me.AboutToolStripMenuItem})
        Me.MenuStrip1.Location = New
System.Drawing.Point(0, 0)
        Me.MenuStrip1.Name = "MenuStrip1"
        Me.MenuStrip1.Size = New
System.Drawing.Size(309, 24)
        Me.MenuStrip1.TabIndex = 7
        Me.MenuStrip1.Text = "MenuStrip1"
        '
        'OpenToolStripMenuItem
        '
        Me.OpenToolStripMenuItem.DropDown
Items.AddRange(New
System.Windows.Forms.ToolStripItem()
{Me.RENDVisualizationToolToolStripMenuIte
m, Me.ReferenceToolStripMenuItem,
Me.HowToUseToolStripMenuItem,
Me.AssessmentToolStripMenuItem})
        Me.OpenToolStripMenuItem.Name =
"OpenToolStripMenuItem"
        Me.OpenToolStripMenuItem.Size =
New System.Drawing.Size(57, 20)
        Me.OpenToolStripMenuItem.Text =
"Open..."
        '
        'RENDVisualizationToolToolStripMe
nuItem
        '
        Me.RENDVisualizationToolToolStrip
MenuItem.Name =
"RENDVisualizationToolToolStripMenuItem"
        Me.RENDVisualizationToolToolStrip
MenuItem.Size = New
System.Drawing.Size(200, 22)
        Me.RENDVisualizationToolToolStrip
MenuItem.Text = "REND Visualization Tool"
        '
        'ReferenceToolStripMenuItem
        '
        Me.ReferenceToolStripMenuItem.Nam
e = "ReferenceToolStripMenuItem"
        Me.ReferenceToolStripMenuItem.Siz
e = New System.Drawing.Size(200, 22)
        Me.ReferenceToolStripMenuItem.Tex
t = "Reference"
        '
        'HowToUseToolStripMenuItem
        '
        Me.HowToUseToolStripMenuItem.Name
= "HowToUseToolStripMenuItem"
        Me.HowToUseToolStripMenuItem.Size
= New System.Drawing.Size(200, 22)
        Me.HowToUseToolStripMenuItem.Text
= "How to Use"
        '
        'AssessmentToolStripMenuItem
        '
        Me.AssessmentToolStripMenuItem.Na
me = "AssessmentToolStripMenuItem"
      Me.AssessmentToolStripMenuItem.Size
= New System.Drawing.Size(200, 22)
        Me.AssessmentToolStripMenuItem.Te
xt = "Assessment"
        '
        'AboutToolStripMenuItem
        '
        Me.AboutToolStripMenuItem.DropDow
nItems.AddRange(New
System.Windows.Forms.ToolStripItem()
{Me.AboutRENDToolStripMenuItem})
        Me.AboutToolStripMenuItem.Name =
"AboutToolStripMenuItem"
        Me.AboutToolStripMenuItem.Size =
New System.Drawing.Size(52, 20)
        Me.AboutToolStripMenuItem.Text =
"About"
        '
        'AboutRENDToolStripMenuItem
        '
        Me.AboutRENDToolStripMenuItem.Nam
e = "AboutRENDToolStripMenuItem"
        Me.AboutRENDToolStripMenuItem.Siz
e = New System.Drawing.Size(140, 22)
        Me.AboutRENDToolStripMenuItem.Tex
t = "About REND"
        '
        'Button1
        '
        Me.Button1.BackColor =
System.Drawing.Color.YellowGreen
        Me.Button1.FlatStyle =
System.Windows.Forms.FlatStyle.Popup
        Me.Button1.Location = New
System.Drawing.Point(56, 51)
        Me.Button1.Name = "Button1"
        Me.Button1.Size = New
System.Drawing.Size(208, 53)
        Me.Button1.TabIndex = 3
        Me.Button1.Text = "Launch REND
Visualization Tool"
        Me.Button1.UseVisualStyleBackColo
r = False
        '
        'frm_Main
        '
        Me.AutoScaleDimensions = New
System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode =
System.Windows.Forms.AutoScaleMode.Font
        Me.BackColor =
System.Drawing.SystemColors.ActiveCaptionText
        Me.ClientSize = New
System.Drawing.Size(309, 306)
        Me.Controls.Add(Me.Button1)
        Me.Controls.Add(Me.Button4)
        Me.Controls.Add(Me.Button3)
        Me.Controls.Add(Me.Button2)
        Me.Controls.Add(Me.MenuStrip1)
        Me.Icon =
CType(resources.GetObject("$this.Icon"),
System.Drawing.Icon)
        Me.MainMenuStrip = Me.MenuStrip1
        Me.Name = "frm_Main"
        Me.StartPosition =
System.Windows.Forms.FormStartPosition.CenterScree
n
        Me.Text = "Main"
        Me.TransparencyKey =
System.Drawing.SystemColors.ActiveBorder
                Me.MenuStrip1.ResumeLayout(False)
        Me.MenuStrip1.PerformLayout()
        Me.ResumeLayout(False)
        Me.PerformLayout()

    End Sub
    Friend WithEvents Button2 As
System.Windows.Forms.Button
    Friend WithEvents Button3 As
System.Windows.Forms.Button
    Friend WithEvents Button4 As
System.Windows.Forms.Button
    Friend WithEvents MenuStrip1 As
System.Windows.Forms.MenuStrip
    Friend WithEvents OpenToolStripMenuItem As
System.Windows.Forms.ToolStripMenuItem
```

```
    Friend WithEvents
RENDVisualizationToolToolStripMenuItem As
System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents ReferenceToolStripMenuItem
As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents HowToUseToolStripMenuItem As
System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents AssessmentToolStripMenuItem
As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents AboutToolStripMenuItem As
System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents AboutRENDToolStripMenuItem
As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents Button1 As
System.Windows.Forms.Button
    Friend WithEvents SaveFileDialog1 As
System.Windows.Forms.SaveFileDialog
End Class
```

Frm_Main.vb

```
Imports System.IO
Imports System.Security.Permissions

Public Class frm_Main
    Private REND As frm_REND = Nothing
    Private run As Boolean = False
    Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
        If Not run Then
            run = True
            REND = New frm_REND(Me)
            REND.Show()
        Else
            MsgBox("Application is running...",
MsgBoxStyle.OkOnly, "REND")
        End If
    End Sub

    Public Property RunningREND()
        Get
            Return run
        End Get
        Set(ByVal value)
            run = value
        End Set
    End Property

    Private Sub Button2_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button2.Click
        Dim r As New frm_Reference
        r.Show()
    End Sub

    Private Sub
RENDVisualizationToolToolStripMenuItem_Click(By
Val sender As System.Object, ByVal e As
System.EventArgs) Handles
RENDVisualizationToolToolStripMenuItem.Click
        Button1_Click(Nothing, Nothing)
    End Sub


    Private Sub
ReferenceToolStripMenuItem_Click(ByVal sender
As System.Object, ByVal e As System.EventArgs)
Handles ReferenceToolStripMenuItem.Click
        Button2_Click(Nothing, Nothing)
    End Sub

    Private Sub
AboutRENDToolStripMenuItem_Click(ByVal sender
As System.Object, ByVal e As System.EventArgs)
Handles AboutRENDToolStripMenuItem.Click
        Dim about As About = New About()
        about.Show()
    End Sub

    Private Sub Button3_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button3.Click
        Dim f As frm_Assessment = New
frm_Assessment
        f.Show()
    End Sub

    Private Sub Button4_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button4.Click
        Dim m As Manual = New Manual
        m.Show()
    End Sub

    Private Sub
HowToUseToolStripMenuItem_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles HowToUseToolStripMenuItem.Click
        Button4_Click(Nothing, Nothing)
    End Sub

    Private Sub
AssessmentToolStripMenuItem_Click(ByVal sender
As System.Object, ByVal e As System.EventArgs)
Handles AssessmentToolStripMenuItem.Click
        Button3_Click(Nothing, Nothing)
    End Sub

    Private Sub frm_Main_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load

    End Sub
End Class
```

Frm_REND.designer.vb

```
<Global.Microsoft.VisualBasic.CompilerServices.
DesignerGenerated()> _
Partial Class frm_REND
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the
component list.
    <System.Diagnostics.DebuggerNonUserCode()>
_
    Protected Overrides Sub Dispose(ByVal
disposing As Boolean)
        Try
            If disposing AndAlso components
IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As
System.ComponentModel.IContainer

    'NOTE: The following procedure is required
by the Windows Form Designer
    'It can be modified using the Windows Form
Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()>
_
    Private Sub InitializeComponent()
        Me.components = New
System.ComponentModel.Container
        Dim DataGridViewCellStyle1 As
System.Windows.Forms.DataGridViewCellStyle =
New System.Windows.Forms.DataGridViewCellStyle
```

```
        Dim DataGridViewCellStyle2 As
System.Windows.Forms.DataGridViewCellStyle =
New System.Windows.Forms.DataGridViewCellStyle
        Dim DataGridViewCellStyle3 As
System.Windows.Forms.DataGridViewCellStyle =
New System.Windows.Forms.DataGridViewCellStyle
        Dim DataGridViewCellStyle4 As
System.Windows.Forms.DataGridViewCellStyle =
New System.Windows.Forms.DataGridViewCellStyle
        Dim DataGridViewCellStyle5 As
System.Windows.Forms.DataGridViewCellStyle =
New System.Windows.Forms.DataGridViewCellStyle
        Dim DataGridViewCellStyle6 As
System.Windows.Forms.DataGridViewCellStyle =
New System.Windows.Forms.DataGridViewCellStyle
        Dim DataGridViewCellStyle7 As
System.Windows.Forms.DataGridViewCellStyle =
New System.Windows.Forms.DataGridViewCellStyle
        Dim DataGridViewCellStyle8 As
System.Windows.Forms.DataGridViewCellStyle =
New System.Windows.Forms.DataGridViewCellStyle
        Dim DataGridViewCellStyle9 As
System.Windows.Forms.DataGridViewCellStyle =
New System.Windows.Forms.DataGridViewCellStyle
        Dim DataGridViewCellStyle10 As
System.Windows.Forms.DataGridViewCellStyle =
New System.Windows.Forms.DataGridViewCellStyle
        Dim DataGridViewCellStyle11 As
System.Windows.Forms.DataGridViewCellStyle =
New System.Windows.Forms.DataGridViewCellStyle
        Dim DataGridViewCellStyle12 As
System.Windows.Forms.DataGridViewCellStyle =
New System.Windows.Forms.DataGridViewCellStyle
        Dim resources As
System.ComponentModel.ComponentResourceManager
= New
System.ComponentModel.ComponentResourceManager(
GetType(frm_REND))
        Me.menuStrip_REND = New
System.Windows.Forms.MenuStrip
        Me.tsm_File = New
System.Windows.Forms.ToolStripMenuItem
        Me.tsm_New = New
System.Windows.Forms.ToolStripMenuItem
        Me.tsm_NewDFA = New
System.Windows.Forms.ToolStripMenuItem
        Me.tsm_NewNFA = New
System.Windows.Forms.ToolStripMenuItem
        Me.tsm_NewENFA = New
System.Windows.Forms.ToolStripMenuItem
        Me.tsm_NewRE = New
System.Windows.Forms.ToolStripMenuItem
        Me.tsm_Exit = New
System.Windows.Forms.ToolStripMenuItem
        Me.tsm_Convert = New
System.Windows.Forms.ToolStripMenuItem
        Me.tsm_ConvertToDFA = New
System.Windows.Forms.ToolStripMenuItem
        Me.tsm_ConvertToRE = New
System.Windows.Forms.ToolStripMenuItem
        Me.tsm_View = New
System.Windows.Forms.ToolStripMenuItem
        Me.tsm_viewTransitionTable = New
System.Windows.Forms.ToolStripMenuItem
        Me.tsm_ViewTransitionFunction = New
System.Windows.Forms.ToolStripMenuItem
        Me.tsm_About = New
System.Windows.Forms.ToolStripMenuItem
        Me.HelpTopicsToolStripMenuItem = New
System.Windows.Forms.ToolStripMenuItem
        Me.AboutRENDToolStripMenuItem = New
System.Windows.Forms.ToolStripMenuItem
        Me.pnl_main = New
System.Windows.Forms.Panel
        Me.tabpages = New
System.Windows.Forms.TabControl
        Me.tab_transdiagram = New
System.Windows.Forms.TabPage
        Me.pnl_diagram = New
System.Windows.Forms.Panel
        Me.Panel1 = New
System.Windows.Forms.Panel
        Me.pnl_Anim = New
System.Windows.Forms.Panel
        Me.tab_transtable = New
System.Windows.Forms.TabPage
        Me.pnl_table = New
System.Windows.Forms.Panel
        Me.Panel3 = New
System.Windows.Forms.Panel
        Me.dgv_orig = New
System.Windows.Forms.DataGridView
        Me.GroupBox1 = New
System.Windows.Forms.GroupBox
        Me.Panel4 = New
System.Windows.Forms.Panel
        Me.GroupBox2 = New
System.Windows.Forms.GroupBox
        Me.DataGridView2 = New
System.Windows.Forms.DataGridView
        Me.Panel5 = New
System.Windows.Forms.Panel
        Me.pnl_pane = New
System.Windows.Forms.Panel
        Me.Panel7 = New
System.Windows.Forms.Panel
        Me.GroupBox5 = New
System.Windows.Forms.GroupBox
        Me.LinkLabel1 = New
System.Windows.Forms.LinkLabel
        Me.RichTextBox4 = New
System.Windows.Forms.RichTextBox
        Me.Panel6 = New
System.Windows.Forms.Panel
        Me.GroupBox3 = New
System.Windows.Forms.GroupBox
        Me.RichTextBox3 = New
System.Windows.Forms.RichTextBox
        Me.pnl_members = New
System.Windows.Forms.Panel
        Me.GroupBox9 = New
System.Windows.Forms.GroupBox
        Me.Button3 = New
System.Windows.Forms.Button
        Me.TextBox1 = New
System.Windows.Forms.TextBox
        Me.pnl_transFunc = New
System.Windows.Forms.Panel
        Me.RichTextBox2 = New
System.Windows.Forms.RichTextBox
        Me.Label2 = New
System.Windows.Forms.Label
        Me.GroupBox7 = New
System.Windows.Forms.GroupBox
        Me.RichTextBox1 = New
System.Windows.Forms.RichTextBox
        Me.GroupBox8 = New
System.Windows.Forms.GroupBox
        Me.pnl_transTable = New
System.Windows.Forms.Panel
        Me.dgv_mini = New
System.Windows.Forms.DataGridView
        Me.GroupBox6 = New
System.Windows.Forms.GroupBox
        Me.Label1 = New
System.Windows.Forms.Label
        Me.pnl_statusAnimate = New
System.Windows.Forms.Panel
        Me.Button2 = New
System.Windows.Forms.Button
        Me.Button1 = New
System.Windows.Forms.Button
        Me.Panel2 = New
System.Windows.Forms.Panel
        Me.Label10 = New
System.Windows.Forms.Label
        Me.Label9 = New
System.Windows.Forms.Label
        Me.Label8 = New
System.Windows.Forms.Label
```

```
        Me.Label7 = New
System.Windows.Forms.Label
        Me.Label6 = New
System.Windows.Forms.Label
        Me.Label4 = New
System.Windows.Forms.Label
        Me.GroupBox4 = New
System.Windows.Forms.GroupBox
        Me.ToolStrip1 = New
System.Windows.Forms.ToolStrip
        Me.ToolStripButton1 = New
System.Windows.Forms.ToolStripButton
        Me.ToolStripButton2 = New
System.Windows.Forms.ToolStripButton
        Me.status_REND = New
System.Windows.Forms.StatusStrip
        Me.tmr_Animate = New
System.Windows.Forms.Timer(Me.components)
        Me.menuStrip_REND.SuspendLayout()
        Me.pnl_main.SuspendLayout()
        Me.tabpages.SuspendLayout()
        Me.tab_transdiagram.SuspendLayout()
        Me.pnl_diagram.SuspendLayout()
        Me.Panel1.SuspendLayout()
        Me.tab_transtable.SuspendLayout()
        Me.pnl_table.SuspendLayout()
        Me.Panel3.SuspendLayout()
        CType(Me.dgv_orig,
System.ComponentModel.ISupportInitialize).Begin
Init()
        Me.Panel4.SuspendLayout()
        Me.GroupBox2.SuspendLayout()
        CType(Me.DataGridView2,
System.ComponentModel.ISupportInitialize).Begin
Init()
        Me.pnl_pane.SuspendLayout()
        Me.Panel7.SuspendLayout()
        Me.GroupBox5.SuspendLayout()
        Me.Panel6.SuspendLayout()
        Me.GroupBox3.SuspendLayout()
        Me.pnl_members.SuspendLayout()
        Me.GroupBox9.SuspendLayout()
        Me.pnl_transFunc.SuspendLayout()
        Me.GroupBox7.SuspendLayout()
        Me.pnl_transTable.SuspendLayout()
        CType(Me.dgv_mini,
System.ComponentModel.ISupportInitialize).Begin
Init()
        Me.GroupBox6.SuspendLayout()
        Me.pnl_statusAnimate.SuspendLayout()
        Me.Panel2.SuspendLayout()
        Me.ToolStrip1.SuspendLayout()
        Me.SuspendLayout()
        '
        'menuStrip_REND
        '
        Me.menuStrip_REND.Font = New
System.Drawing.Font("Arial", 9.75!,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.menuStrip_REND.Items.AddRange(New
System.Windows.Forms.ToolStripItem()
{Me.tsm_File, Me.tsm_Convert, Me.tsm_View,
Me.tsm_About})
        Me.menuStrip_REND.Location = New
System.Drawing.Point(0, 0)
        Me.menuStrip_REND.Name =
"menuStrip_REND"
        Me.menuStrip_REND.Size = New
System.Drawing.Size(1276, 24)
        Me.menuStrip_REND.TabIndex = 0
        Me.menuStrip_REND.Text = "MenuStrip1"
        '
        'tsm_File
        '
        Me.tsm_File.DropDownItems.AddRange(New
System.Windows.Forms.ToolStripItem()
{Me.tsm_New, Me.tsm_Exit})
        Me.tsm_File.Font = New
System.Drawing.Font("Arial", 9.75!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.tsm_File.Name = "tsm_File"
        Me.tsm_File.ShortcutKeys =
CType((System.Windows.Forms.Keys.Alt Or
System.Windows.Forms.Keys.F),
System.Windows.Forms.Keys)
        Me.tsm_File.Size = New
System.Drawing.Size(41, 20)
        Me.tsm_File.Text = "File"
        '
        'tsm_New
        '
        Me.tsm_New.DropDownItems.AddRange(New
System.Windows.Forms.ToolStripItem()
{Me.tsm_NewDFA, Me.tsm_NewNFA, Me.tsm_NewENFA,
Me.tsm_NewRE})
        Me.tsm_New.Image =
Global.SpecialProject.My.Resources.Resources.NF
A
        Me.tsm_New.Name = "tsm_New"
        Me.tsm_New.ShortcutKeys =
CType((System.Windows.Forms.Keys.Control Or
System.Windows.Forms.Keys.N),
System.Windows.Forms.Keys)
        Me.tsm_New.Size = New
System.Drawing.Size(158, 22)
        Me.tsm_New.Text = "New..."
        '
        'tsm_NewDFA
        '
        Me.tsm_NewDFA.Image =
Global.SpecialProject.My.Resources.Resources.DF
A
        Me.tsm_NewDFA.Name = "tsm_NewDFA"
        Me.tsm_NewDFA.ShortcutKeyDisplayString
= ""
        Me.tsm_NewDFA.ShortcutKeys =
CType((System.Windows.Forms.Keys.Control Or
System.Windows.Forms.Keys.D),
System.Windows.Forms.Keys)
        Me.tsm_NewDFA.Size = New
System.Drawing.Size(378, 22)
        Me.tsm_NewDFA.Text = "Deterministic
Finite Automaton"
        '
        'tsm_NewNFA
        '
        Me.tsm_NewNFA.Image =
Global.SpecialProject.My.Resources.Resources.NF
A
        Me.tsm_NewNFA.Name = "tsm_NewNFA"
        Me.tsm_NewNFA.ShortcutKeys =
CType((System.Windows.Forms.Keys.Control Or
System.Windows.Forms.Keys.N),
System.Windows.Forms.Keys)
        Me.tsm_NewNFA.Size = New
System.Drawing.Size(378, 22)
        Me.tsm_NewNFA.Text = "Non-Deterministic
Finite Automaton"
        '
        'tsm_NewENFA
        '
        Me.tsm_NewENFA.Image =
Global.SpecialProject.My.Resources.Resources.NF
A_E1
        Me.tsm_NewENFA.Name = "tsm_NewENFA"
        Me.tsm_NewENFA.ShortcutKeys =
CType((System.Windows.Forms.Keys.Control Or
System.Windows.Forms.Keys.E),
System.Windows.Forms.Keys)
        Me.tsm_NewENFA.Size = New
System.Drawing.Size(378, 22)
        Me.tsm_NewENFA.Text = "Non-
Deterministic Finite Automaton- Epsilon"
        '
        'tsm_NewRE
        '
        Me.tsm_NewRE.Image =
Global.SpecialProject.My.Resources.Resources.RE
```

```
        Me.tsm_NewRE.Name = "tsm_NewRE"
        Me.tsm_NewRE.ShortcutKeys =
CType((System.Windows.Forms.Keys.Control Or
System.Windows.Forms.Keys.R),
System.Windows.Forms.Keys)
        Me.tsm_NewRE.Size = New
System.Drawing.Size(378, 22)
        Me.tsm_NewRE.Text = "Regular
Expression"
        '
        'tsm_Exit
        '
        Me.tsm_Exit.Name = "tsm_Exit"
        Me.tsm_Exit.Size = New
System.Drawing.Size(158, 22)
        Me.tsm_Exit.Text = "Exit"
        '
        'tsm_Convert
        '
        Me.tsm_Convert.DropDownItems.AddRange(N
ew System.Windows.Forms.ToolStripItem()
{Me.tsm_ConvertToDFA, Me.tsm_ConvertToRE})
        Me.tsm_Convert.Font = New
System.Drawing.Font("Arial", 9.75!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.tsm_Convert.Name = "tsm_Convert"
        Me.tsm_Convert.ShortcutKeys =
CType((System.Windows.Forms.Keys.Alt Or
System.Windows.Forms.Keys.C),
System.Windows.Forms.Keys)
        Me.tsm_Convert.Size = New
System.Drawing.Size(78, 20)
        Me.tsm_Convert.Text = "Convert to"
        '
        'tsm_ConvertToDFA
        '
        Me.tsm_ConvertToDFA.Image =
Global.SpecialProject.My.Resources.Resources.DF
A
        Me.tsm_ConvertToDFA.Name =
"tsm_ConvertToDFA"
        Me.tsm_ConvertToDFA.Size = New
System.Drawing.Size(189, 22)
        Me.tsm_ConvertToDFA.Text = "DFA"
        '
        'tsm_ConvertToRE
        '
        Me.tsm_ConvertToRE.Image =
Global.SpecialProject.My.Resources.Resources.RE
        Me.tsm_ConvertToRE.Name =
"tsm_ConvertToRE"
        Me.tsm_ConvertToRE.Size = New
System.Drawing.Size(189, 22)
        Me.tsm_ConvertToRE.Text = "Regular
Expression"
        '
        'tsm_View
        '
        Me.tsm_View.DropDownItems.AddRange(New
System.Windows.Forms.ToolStripItem()
{Me.tsm_viewTransitionTable,
Me.tsm_ViewTransitionFunction})
        Me.tsm_View.Font = New
System.Drawing.Font("Arial", 9.75!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.tsm_View.Name = "tsm_View"
        Me.tsm_View.ShortcutKeys =
CType((System.Windows.Forms.Keys.Alt Or
System.Windows.Forms.Keys.V),
System.Windows.Forms.Keys)
        Me.tsm_View.Size = New
System.Drawing.Size(48, 20)
        Me.tsm_View.Text = "View"
        '
        'tsm_viewTransitionTable
        '
        Me.tsm_viewTransitionTable.Image =
Global.SpecialProject.My.Resources.Resources.vi
ewtable
        Me.tsm_viewTransitionTable.Name =
"tsm_viewTransitionTable"
        Me.tsm_viewTransitionTable.ShortcutKeys
= CType(((System.Windows.Forms.Keys.Control Or
System.Windows.Forms.Keys.Shift) _
                    Or
System.Windows.Forms.Keys.T),
System.Windows.Forms.Keys)
        Me.tsm_viewTransitionTable.Size = New
System.Drawing.Size(263, 22)
        Me.tsm_viewTransitionTable.Text =
"Transition Table"
        '
        'tsm_ViewTransitionFunction
        '
        Me.tsm_ViewTransitionFunction.Image =
Global.SpecialProject.My.Resources.Resources.vi
ewdiagram
        Me.tsm_ViewTransitionFunction.Name =
"tsm_ViewTransitionFunction"
        Me.tsm_ViewTransitionFunction.ShortcutK
eys = CType(((System.Windows.Forms.Keys.Control
Or System.Windows.Forms.Keys.Shift) _
                    Or
System.Windows.Forms.Keys.D),
System.Windows.Forms.Keys)
        Me.tsm_ViewTransitionFunction.Size =
New System.Drawing.Size(263, 22)
        Me.tsm_ViewTransitionFunction.Text =
"Transition Diagram"
        '
        'tsm_About
        '
        Me.tsm_About.DropDownItems.AddRange(New
System.Windows.Forms.ToolStripItem()
{Me.HelpTopicsToolStripMenuItem,
Me.AboutRENDToolStripMenuItem})
        Me.tsm_About.Font = New
System.Drawing.Font("Arial", 9.75!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.tsm_About.Name = "tsm_About"
        Me.tsm_About.ShortcutKeys =
CType((System.Windows.Forms.Keys.Alt Or
System.Windows.Forms.Keys.A),
System.Windows.Forms.Keys)
        Me.tsm_About.Size = New
System.Drawing.Size(54, 20)
        Me.tsm_About.Text = "About"
        '
        'HelpTopicsToolStripMenuItem
        '
        Me.HelpTopicsToolStripMenuItem.Image =
Global.SpecialProject.My.Resources.Resources.he
lp
        Me.HelpTopicsToolStripMenuItem.Name =
"HelpTopicsToolStripMenuItem"
        Me.HelpTopicsToolStripMenuItem.Shortcut
Keys = CType((System.Windows.Forms.Keys.Shift
Or System.Windows.Forms.Keys.F1),
System.Windows.Forms.Keys)
        Me.HelpTopicsToolStripMenuItem.Size =
New System.Drawing.Size(198, 22)
        Me.HelpTopicsToolStripMenuItem.Text =
"Help topics"
        '
        'AboutRENDToolStripMenuItem
        '
        Me.AboutRENDToolStripMenuItem.Name =
"AboutRENDToolStripMenuItem"
        Me.AboutRENDToolStripMenuItem.Size =
New System.Drawing.Size(198, 22)
        Me.AboutRENDToolStripMenuItem.Text =
"About REND"
        '
        'pnl_main
        '
        Me.pnl_main.Anchor =
CType((((System.Windows.Forms.AnchorStyles.Top
Or System.Windows.Forms.AnchorStyles.Bottom) _
                    Or
System.Windows.Forms.AnchorStyles.Left) _
                    Or
System.Windows.Forms.AnchorStyles.Right),
System.Windows.Forms.AnchorStyles)
        Me.pnl_main.BackColor =
System.Drawing.SystemColors.ActiveBorder
        Me.pnl_main.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle
        Me.pnl_main.Controls.Add(Me.tabpages)
        Me.pnl_main.Controls.Add(Me.pnl_pane)
        Me.pnl_main.ImeMode =
System.Windows.Forms.ImeMode.Off
        Me.pnl_main.Location = New
System.Drawing.Point(12, 34)
        Me.pnl_main.Name = "pnl_main"
        Me.pnl_main.Size = New
System.Drawing.Size(1252, 681)
        Me.pnl_main.TabIndex = 1
        '
        'tabpages
        '
        Me.tabpages.Controls.Add(Me.tab_transdi
agram)
        Me.tabpages.Controls.Add(Me.tab_transta
ble)
        Me.tabpages.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.tabpages.Location = New
System.Drawing.Point(-1, 49)
        Me.tabpages.Name = "tabpages"
        Me.tabpages.SelectedIndex = 0
        Me.tabpages.Size = New
System.Drawing.Size(1010, 527)
        Me.tabpages.TabIndex = 0
        '
        'tab_transdiagram
        '
        Me.tab_transdiagram.Controls.Add(Me.pnl
_diagram)
        Me.tab_transdiagram.Location = New
System.Drawing.Point(4, 23)
        Me.tab_transdiagram.Name =
"tab_transdiagram"
        Me.tab_transdiagram.Padding = New
System.Windows.Forms.Padding(3)
        Me.tab_transdiagram.Size = New
System.Drawing.Size(1002, 500)
        Me.tab_transdiagram.TabIndex = 0
        Me.tab_transdiagram.Text = "Transition
Diagram"
        Me.tab_transdiagram.UseVisualStyleBackC
olor = True
        '
        'pnl_diagram
        '
        Me.pnl_diagram.AutoSize = True
        Me.pnl_diagram.AutoSizeMode =
System.Windows.Forms.AutoSizeMode.GrowAndShrink
        Me.pnl_diagram.BackColor =
System.Drawing.Color.YellowGreen
        Me.pnl_diagram.BorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D
        Me.pnl_diagram.Controls.Add(Me.Panel1)
        Me.pnl_diagram.Location = New
System.Drawing.Point(0, 2)
        Me.pnl_diagram.Name = "pnl_diagram"
        Me.pnl_diagram.Size = New
System.Drawing.Size(1002, 500)
        Me.pnl_diagram.TabIndex = 0
        '
        'Panel1
        '
        Me.Panel1.BackColor =
System.Drawing.SystemColors.ActiveCaptionText
        Me.Panel1.Controls.Add(Me.pnl_Anim)
        Me.Panel1.Location = New
System.Drawing.Point(3, 3)
        Me.Panel1.Name = "Panel1"
        Me.Panel1.Size = New
System.Drawing.Size(992, 490)
        Me.Panel1.TabIndex = 0
        '
        'pnl_Anim
        '
        Me.pnl_Anim.BackColor =
System.Drawing.SystemColors.ActiveCaptionText
        Me.pnl_Anim.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Italic,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.pnl_Anim.Location = New
System.Drawing.Point(1, 4)
        Me.pnl_Anim.Name = "pnl_Anim"
        Me.pnl_Anim.Size = New
System.Drawing.Size(988, 490)
        Me.pnl_Anim.TabIndex = 0
        Me.pnl_Anim.Visible = False
        '
        'tab_transtable
        '
        Me.tab_transtable.Controls.Add(Me.pnl_t
able)
        Me.tab_transtable.ForeColor =
System.Drawing.SystemColors.InactiveCaption
        Me.tab_transtable.Location = New
System.Drawing.Point(4, 23)
        Me.tab_transtable.Name =
"tab_transtable"
        Me.tab_transtable.Padding = New
System.Windows.Forms.Padding(3)
        Me.tab_transtable.Size = New
System.Drawing.Size(1002, 500)
        Me.tab_transtable.TabIndex = 1
        Me.tab_transtable.Text = "Transition
Table"
        Me.tab_transtable.UseVisualStyleBackCol
or = True
        '
        'pnl_table
        '
        Me.pnl_table.BackColor =
System.Drawing.Color.LightGreen
        Me.pnl_table.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle
        Me.pnl_table.Controls.Add(Me.Panel3)
        Me.pnl_table.Controls.Add(Me.Panel4)
        Me.pnl_table.Controls.Add(Me.Panel5)
        Me.pnl_table.Location = New
System.Drawing.Point(3, 3)
        Me.pnl_table.Name = "pnl_table"
        Me.pnl_table.Size = New
System.Drawing.Size(998, 497)
        Me.pnl_table.TabIndex = 0
        '
        'Panel3
        '
        Me.Panel3.BackColor =
System.Drawing.SystemColors.ButtonFace
        Me.Panel3.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle
        Me.Panel3.Controls.Add(Me.dgv_orig)
        Me.Panel3.Controls.Add(Me.GroupBox1)
        Me.Panel3.Location = New
System.Drawing.Point(22, 22)
        Me.Panel3.Name = "Panel3"
        Me.Panel3.Size = New
System.Drawing.Size(953, 217)
        Me.Panel3.TabIndex = 3
        '
        'dgv_orig
        '
        Me.dgv_orig.AllowUserToAddRows = False
        Me.dgv_orig.AllowUserToDeleteRows =
False
```

```
        Me.dgv_orig.AllowUserToResizeRows =
False
        Me.dgv_orig.AutoSizeColumnsMode =
System.Windows.Forms.DataGridViewAutoSizeColumn
sMode.AllCells
        Me.dgv_orig.AutoSizeRowsMode =
System.Windows.Forms.DataGridViewAutoSizeRowsMo
de.AllCells
        Me.dgv_orig.BackgroundColor =
System.Drawing.Color.DarkSeaGreen
        DataGridViewCellStyle1.Alignment =
System.Windows.Forms.DataGridViewContentAlignme
nt.MiddleCenter
        DataGridViewCellStyle1.BackColor =
System.Drawing.SystemColors.Control
        DataGridViewCellStyle1.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        DataGridViewCellStyle1.ForeColor =
System.Drawing.SystemColors.WindowText
        DataGridViewCellStyle1.SelectionBackCol
or = System.Drawing.SystemColors.Highlight
        DataGridViewCellStyle1.SelectionForeCol
or = System.Drawing.SystemColors.HighlightText
        DataGridViewCellStyle1.WrapMode =
System.Windows.Forms.DataGridViewTriState.
[True]
        Me.dgv_orig.ColumnHeadersDefaultCellSty
le = DataGridViewCellStyle1
        Me.dgv_orig.ColumnHeadersHeightSizeMode
=
System.Windows.Forms.DataGridViewColumnHeadersH
eightSizeMode.DisableResizing
        DataGridViewCellStyle2.Alignment =
System.Windows.Forms.DataGridViewContentAlignme
nt.MiddleCenter
        DataGridViewCellStyle2.BackColor =
System.Drawing.SystemColors.Window
        DataGridViewCellStyle2.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        DataGridViewCellStyle2.ForeColor =
System.Drawing.SystemColors.InactiveCaption
        DataGridViewCellStyle2.SelectionBackCol
or = System.Drawing.SystemColors.Highlight
        DataGridViewCellStyle2.SelectionForeCol
or = System.Drawing.SystemColors.HighlightText
        DataGridViewCellStyle2.WrapMode =
System.Windows.Forms.DataGridViewTriState.
[False]
        Me.dgv_orig.DefaultCellStyle =
DataGridViewCellStyle2
        Me.dgv_orig.GridColor =
System.Drawing.Color.DarkSeaGreen
        Me.dgv_orig.Location = New
System.Drawing.Point(10, 23)
        Me.dgv_orig.MultiSelect = False
        Me.dgv_orig.Name = "dgv_orig"
        Me.dgv_orig.ReadOnly = True
        DataGridViewCellStyle3.Alignment =
System.Windows.Forms.DataGridViewContentAlignme
nt.MiddleCenter
        DataGridViewCellStyle3.BackColor =
System.Drawing.SystemColors.Control
        DataGridViewCellStyle3.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        DataGridViewCellStyle3.ForeColor =
System.Drawing.SystemColors.WindowText
        DataGridViewCellStyle3.SelectionBackCol
or = System.Drawing.SystemColors.Highlight
        DataGridViewCellStyle3.SelectionForeCol
or = System.Drawing.SystemColors.HighlightText
        DataGridViewCellStyle3.WrapMode =
System.Windows.Forms.DataGridViewTriState.
[True]
        Me.dgv_orig.RowHeadersDefaultCellStyle
= DataGridViewCellStyle3
        Me.dgv_orig.RowHeadersWidthSizeMode =
System.Windows.Forms.DataGridViewRowHeadersWidt
hSizeMode.AutoSizeToDisplayedHeaders
        DataGridViewCellStyle4.Alignment =
System.Windows.Forms.DataGridViewContentAlignme
nt.MiddleCenter
        DataGridViewCellStyle4.BackColor =
System.Drawing.Color.White
        DataGridViewCellStyle4.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        DataGridViewCellStyle4.ForeColor =
System.Drawing.Color.Black
        DataGridViewCellStyle4.SelectionBackCol
or = System.Drawing.SystemColors.Highlight
        Me.dgv_orig.RowsDefaultCellStyle =
DataGridViewCellStyle4
        Me.dgv_orig.SelectionMode =
System.Windows.Forms.DataGridViewSelectionMode.
CellSelect
        Me.dgv_orig.Size = New
System.Drawing.Size(931, 177)
        Me.dgv_orig.TabIndex = 1
        '
        'GroupBox1
        '
        Me.GroupBox1.Location = New
System.Drawing.Point(3, 4)
        Me.GroupBox1.Name = "GroupBox1"
        Me.GroupBox1.Size = New
System.Drawing.Size(945, 208)
        Me.GroupBox1.TabIndex = 6
        Me.GroupBox1.TabStop = False
        Me.GroupBox1.Text = "Original
Transition Table"
        '
        'Panel4
        '
        Me.Panel4.BackColor =
System.Drawing.SystemColors.ButtonFace
        Me.Panel4.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle
        Me.Panel4.Controls.Add(Me.GroupBox2)
        Me.Panel4.Location = New
System.Drawing.Point(22, 257)
        Me.Panel4.Name = "Panel4"
        Me.Panel4.Size = New
System.Drawing.Size(953, 217)
        Me.Panel4.TabIndex = 4
        '
        'GroupBox2
        '
        Me.GroupBox2.Controls.Add(Me.DataGridVi
ew2)
        Me.GroupBox2.Location = New
System.Drawing.Point(3, 8)
        Me.GroupBox2.Name = "GroupBox2"
        Me.GroupBox2.Size = New
System.Drawing.Size(945, 204)
        Me.GroupBox2.TabIndex = 1
        Me.GroupBox2.TabStop = False
        Me.GroupBox2.Text = "Converted
Transition Table"
        '
        'DataGridView2
        '
        Me.DataGridView2.AllowUserToAddRows =
False
        Me.DataGridView2.AllowUserToDeleteRows
= False
        Me.DataGridView2.AllowUserToResizeRows
= False
```

```
        Me.DataGridView2.AutoSizeColumnsMode =
System.Windows.Forms.DataGridViewAutoSizeColumn
sMode.AllCells
        Me.DataGridView2.AutoSizeRowsMode =
System.Windows.Forms.DataGridViewAutoSizeRowsMo
de.AllCells
        Me.DataGridView2.BackgroundColor =
System.Drawing.Color.DarkSeaGreen
        DataGridViewCellStyle5.Alignment =
System.Windows.Forms.DataGridViewContentAlignme
nt.MiddleCenter
        DataGridViewCellStyle5.BackColor =
System.Drawing.SystemColors.Control
        DataGridViewCellStyle5.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        DataGridViewCellStyle5.ForeColor =
System.Drawing.SystemColors.WindowText
        DataGridViewCellStyle5.SelectionBackCol
or = System.Drawing.SystemColors.Highlight
        DataGridViewCellStyle5.SelectionForeCol
or = System.Drawing.SystemColors.HighlightText
        DataGridViewCellStyle5.WrapMode =
System.Windows.Forms.DataGridViewTriState.
[True]
        Me.DataGridView2.ColumnHeadersDefaultCe
llStyle = DataGridViewCellStyle5
        Me.DataGridView2.ColumnHeadersHeightSiz
eMode =
System.Windows.Forms.DataGridViewColumnHeadersH
eightSizeMode.DisableResizing
        DataGridViewCellStyle6.Alignment =
System.Windows.Forms.DataGridViewContentAlignme
nt.MiddleCenter
        DataGridViewCellStyle6.BackColor =
System.Drawing.SystemColors.Window
        DataGridViewCellStyle6.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        DataGridViewCellStyle6.ForeColor =
System.Drawing.SystemColors.InactiveCaption
        DataGridViewCellStyle6.SelectionBackCol
or = System.Drawing.SystemColors.Highlight
        DataGridViewCellStyle6.SelectionForeCol
or = System.Drawing.SystemColors.HighlightText
        DataGridViewCellStyle6.WrapMode =
System.Windows.Forms.DataGridViewTriState.
[False]
        Me.DataGridView2.DefaultCellStyle =
DataGridViewCellStyle6
        Me.DataGridView2.GridColor =
System.Drawing.Color.DarkSeaGreen
        Me.DataGridView2.Location = New
System.Drawing.Point(8, 19)
        Me.DataGridView2.MultiSelect = False
        Me.DataGridView2.Name = "DataGridView2"
        Me.DataGridView2.ReadOnly = True
        DataGridViewCellStyle7.Alignment =
System.Windows.Forms.DataGridViewContentAlignme
nt.MiddleCenter
        DataGridViewCellStyle7.BackColor =
System.Drawing.SystemColors.Control
        DataGridViewCellStyle7.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        DataGridViewCellStyle7.ForeColor =
System.Drawing.SystemColors.WindowText
        DataGridViewCellStyle7.SelectionBackCol
or = System.Drawing.Color.DarkSeaGreen
        DataGridViewCellStyle7.SelectionForeCol
or = System.Drawing.SystemColors.HighlightText
        DataGridViewCellStyle7.WrapMode =
System.Windows.Forms.DataGridViewTriState.
[True]
        Me.DataGridView2.RowHeadersDefaultCellS
tyle = DataGridViewCellStyle7
        Me.DataGridView2.RowHeadersWidthSizeMod
e =
System.Windows.Forms.DataGridViewRowHeadersWidt
hSizeMode.AutoSizeToDisplayedHeaders
        DataGridViewCellStyle8.Alignment =
System.Windows.Forms.DataGridViewContentAlignme
nt.MiddleCenter
        DataGridViewCellStyle8.BackColor =
System.Drawing.Color.White
        DataGridViewCellStyle8.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        DataGridViewCellStyle8.ForeColor =
System.Drawing.Color.Black
        DataGridViewCellStyle8.SelectionBackCol
or = System.Drawing.SystemColors.Highlight
        Me.DataGridView2.RowsDefaultCellStyle =
DataGridViewCellStyle8
        Me.DataGridView2.SelectionMode =
System.Windows.Forms.DataGridViewSelectionMode.
CellSelect
        Me.DataGridView2.Size = New
System.Drawing.Size(930, 177)
        Me.DataGridView2.TabIndex = 2
        '
        'Panel5
        '
        Me.Panel5.BackColor =
System.Drawing.SystemColors.ActiveBorder
        Me.Panel5.BorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D
        Me.Panel5.Location = New
System.Drawing.Point(4, 3)
        Me.Panel5.Name = "Panel5"
        Me.Panel5.Size = New
System.Drawing.Size(989, 489)
        Me.Panel5.TabIndex = 1
        '
        'pnl_pane
        '
        Me.pnl_pane.Anchor =
CType((((System.Windows.Forms.AnchorStyles.Top
Or System.Windows.Forms.AnchorStyles.Bottom) _
                    Or
System.Windows.Forms.AnchorStyles.Left) _
                    Or
System.Windows.Forms.AnchorStyles.Right),
System.Windows.Forms.AnchorStyles)
        Me.pnl_pane.AutoSizeMode =
System.Windows.Forms.AutoSizeMode.GrowAndShrink
        Me.pnl_pane.BackColor =
System.Drawing.SystemColors.Info
        Me.pnl_pane.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle
        Me.pnl_pane.Controls.Add(Me.Panel7)
        Me.pnl_pane.Controls.Add(Me.Panel6)
        Me.pnl_pane.Controls.Add(Me.pnl_members
)
        Me.pnl_pane.Controls.Add(Me.pnl_transFu
nc)
        Me.pnl_pane.Controls.Add(Me.pnl_transTa
ble)
        Me.pnl_pane.Controls.Add(Me.pnl_statusA
nimate)
        Me.pnl_pane.Controls.Add(Me.Panel2)
        Me.pnl_pane.Location = New
System.Drawing.Point(-1, 70)
        Me.pnl_pane.Name = "pnl_pane"
        Me.pnl_pane.Size = New
System.Drawing.Size(1252, 608)
        Me.pnl_pane.TabIndex = 0
        '
        'Panel7
        '
        Me.Panel7.BackColor =
System.Drawing.Color.DarkSeaGreen
```

```
        Me.Panel7.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle
        Me.Panel7.Controls.Add(Me.GroupBox5)
        Me.Panel7.Location = New
System.Drawing.Point(435, 510)
        Me.Panel7.Name = "Panel7"
        Me.Panel7.Size = New
System.Drawing.Size(374, 89)
        Me.Panel7.TabIndex = 5
        '
        'GroupBox5
        '
        Me.GroupBox5.Controls.Add(Me.LinkLabel1
)
        Me.GroupBox5.Controls.Add(Me.RichTextBo
x4)
        Me.GroupBox5.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.GroupBox5.ForeColor =
System.Drawing.Color.Navy
        Me.GroupBox5.Location = New
System.Drawing.Point(3, 3)
        Me.GroupBox5.Name = "GroupBox5"
        Me.GroupBox5.Size = New
System.Drawing.Size(366, 81)
        Me.GroupBox5.TabIndex = 0
        Me.GroupBox5.TabStop = False
        Me.GroupBox5.Text = "About
Deterministic Finite Automaton"
        '
        'LinkLabel1
        '
        Me.LinkLabel1.AutoSize = True
        Me.LinkLabel1.Location = New
System.Drawing.Point(318, 68)
        Me.LinkLabel1.Name = "LinkLabel1"
        Me.LinkLabel1.Size = New
System.Drawing.Size(45, 14)
        Me.LinkLabel1.TabIndex = 1
        Me.LinkLabel1.TabStop = True
        Me.LinkLabel1.Text = "More..."
        '
        'RichTextBox4
        '
        Me.RichTextBox4.BackColor =
System.Drawing.SystemColors.GradientInactiveCap
tion
        Me.RichTextBox4.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.RichTextBox4.Location = New
System.Drawing.Point(6, 18)
        Me.RichTextBox4.MaxLength = 62
        Me.RichTextBox4.Name = "RichTextBox4"
        Me.RichTextBox4.Size = New
System.Drawing.Size(354, 54)
        Me.RichTextBox4.TabIndex = 0
        Me.RichTextBox4.Text = ""
        '
        'Panel6
        '
        Me.Panel6.BackColor =
System.Drawing.Color.DarkSeaGreen
        Me.Panel6.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle
        Me.Panel6.Controls.Add(Me.GroupBox3)
        Me.Panel6.Location = New
System.Drawing.Point(188, 509)
        Me.Panel6.Name = "Panel6"
        Me.Panel6.Size = New
System.Drawing.Size(241, 90)
        Me.Panel6.TabIndex = 4
        '
        'GroupBox3
        '
        Me.GroupBox3.AutoSizeMode =
System.Windows.Forms.AutoSizeMode.GrowAndShrink
        Me.GroupBox3.Controls.Add(Me.RichTextBo
x3)
        Me.GroupBox3.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.GroupBox3.ForeColor =
System.Drawing.Color.Crimson
        Me.GroupBox3.Location = New
System.Drawing.Point(3, 3)
        Me.GroupBox3.Name = "GroupBox3"
        Me.GroupBox3.Size = New
System.Drawing.Size(233, 82)
        Me.GroupBox3.TabIndex = 2
        Me.GroupBox3.TabStop = False
        Me.GroupBox3.Text = "State Equivalence"
        '
        'RichTextBox3
        '
        Me.RichTextBox3.BackColor =
System.Drawing.Color.DarkSeaGreen
        Me.RichTextBox3.BorderStyle =
System.Windows.Forms.BorderStyle.None
        Me.RichTextBox3.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.RichTextBox3.Location = New
System.Drawing.Point(14, 19)
        Me.RichTextBox3.Name = "RichTextBox3"
        Me.RichTextBox3.ReadOnly = True
        Me.RichTextBox3.Size = New
System.Drawing.Size(213, 54)
        Me.RichTextBox3.TabIndex = 0
        Me.RichTextBox3.Text = "" &
Global.Microsoft.VisualBasic.ChrW(10)
        '
        'pnl_members
        '
        Me.pnl_members.Anchor =
CType((System.Windows.Forms.AnchorStyles.Top Or
System.Windows.Forms.AnchorStyles.Right),
System.Windows.Forms.AnchorStyles)
        Me.pnl_members.BackColor =
System.Drawing.Color.DarkSeaGreen
        Me.pnl_members.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle
        Me.pnl_members.Controls.Add(Me.GroupBox
9)
        Me.pnl_members.Location = New
System.Drawing.Point(1011, 493)
        Me.pnl_members.Name = "pnl_members"
        Me.pnl_members.Size = New
System.Drawing.Size(231, 106)
        Me.pnl_members.TabIndex = 2
        '
        'GroupBox9
        '
        Me.GroupBox9.Controls.Add(Me.Button3)
        Me.GroupBox9.Controls.Add(Me.TextBox1)
        Me.GroupBox9.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.GroupBox9.ForeColor =
System.Drawing.Color.Red
        Me.GroupBox9.Location = New
System.Drawing.Point(3, 0)
        Me.GroupBox9.Name = "GroupBox9"
        Me.GroupBox9.Size = New
System.Drawing.Size(218, 100)
        Me.GroupBox9.TabIndex = 1
        Me.GroupBox9.TabStop = False
        Me.GroupBox9.Text = "Try A Binary
String Here!"
        '
        'Button3
        '
        Me.Button3.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.Button3.ForeColor =
System.Drawing.SystemColors.ControlText
        Me.Button3.Location = New
System.Drawing.Point(53, 57)
        Me.Button3.Name = "Button3"
        Me.Button3.Size = New
System.Drawing.Size(115, 30)
        Me.Button3.TabIndex = 1
        Me.Button3.Text = "Test Membership"
        Me.Button3.UseVisualStyleBackColor =
True
        '
        'TextBox1
        '
        Me.TextBox1.Location = New
System.Drawing.Point(31, 29)
        Me.TextBox1.MaxLength = 25
        Me.TextBox1.Name = "TextBox1"
        Me.TextBox1.Size = New
System.Drawing.Size(158, 20)
        Me.TextBox1.TabIndex = 0
        '
        'pnl_transFunc
        '
        Me.pnl_transFunc.Anchor =
CType((System.Windows.Forms.AnchorStyles.Top Or
System.Windows.Forms.AnchorStyles.Right),
System.Windows.Forms.AnchorStyles)
        Me.pnl_transFunc.BackColor =
System.Drawing.Color.DarkSeaGreen
        Me.pnl_transFunc.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle
        Me.pnl_transFunc.Controls.Add(Me.RichTe
xtBox2)
        Me.pnl_transFunc.Controls.Add(Me.Label2
)
        Me.pnl_transFunc.Controls.Add(Me.GroupB
ox7)
        Me.pnl_transFunc.Controls.Add(Me.GroupB
ox8)
        Me.pnl_transFunc.ForeColor =
System.Drawing.SystemColors.ControlText
        Me.pnl_transFunc.Location = New
System.Drawing.Point(1011, 242)
        Me.pnl_transFunc.Name = "pnl_transFunc"
        Me.pnl_transFunc.Size = New
System.Drawing.Size(232, 247)
        Me.pnl_transFunc.TabIndex = 1
        '
        'RichTextBox2
        '
        Me.RichTextBox2.Cursor =
System.Windows.Forms.Cursors.Hand
        Me.RichTextBox2.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.RichTextBox2.Location = New
System.Drawing.Point(7, 191)
        Me.RichTextBox2.MaxLength = 225
        Me.RichTextBox2.Name = "RichTextBox2"
        Me.RichTextBox2.ReadOnly = True
        Me.RichTextBox2.Size = New
System.Drawing.Size(216, 47)
        Me.RichTextBox2.TabIndex = 5
        Me.RichTextBox2.Text = ""
        '
        'Label2
        '
        Me.Label2.BackColor =
System.Drawing.Color.DarkSeaGreen
        Me.Label2.FlatStyle =
System.Windows.Forms.FlatStyle.Flat
        Me.Label2.Font = New
System.Drawing.Font("Arial", 9.0!,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.Label2.ForeColor =
System.Drawing.SystemColors.ControlText
        Me.Label2.Location = New
System.Drawing.Point(40, 174)
        Me.Label2.Name = "Label2"
        Me.Label2.Size = New
System.Drawing.Size(143, 17)
        Me.Label2.TabIndex = 4
        Me.Label2.Text = "Result On Conversion"
        Me.Label2.TextAlign =
System.Drawing.ContentAlignment.MiddleCenter
        '
        'GroupBox7
        '
        Me.GroupBox7.Controls.Add(Me.RichTextBo
x1)
        Me.GroupBox7.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.GroupBox7.Location = New
System.Drawing.Point(2, 3)
        Me.GroupBox7.Name = "GroupBox7"
        Me.GroupBox7.Size = New
System.Drawing.Size(224, 168)
        Me.GroupBox7.TabIndex = 6
        Me.GroupBox7.TabStop = False
        Me.GroupBox7.Text = "Logs"
        '
        'RichTextBox1
        '
        Me.RichTextBox1.AutoWordSelection =
True
        Me.RichTextBox1.BackColor =
System.Drawing.Color.LightCyan
        Me.RichTextBox1.Cursor =
System.Windows.Forms.Cursors.Hand
        Me.RichTextBox1.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.RichTextBox1.Location = New
System.Drawing.Point(5, 15)
        Me.RichTextBox1.Name = "RichTextBox1"
        Me.RichTextBox1.ReadOnly = True
        Me.RichTextBox1.Size = New
System.Drawing.Size(212, 146)
        Me.RichTextBox1.TabIndex = 2
        Me.RichTextBox1.Text = ""
        '
        'GroupBox8
        '
        Me.GroupBox8.Location = New
System.Drawing.Point(3, 177)
        Me.GroupBox8.Name = "GroupBox8"
        Me.GroupBox8.Size = New
System.Drawing.Size(226, 67)
        Me.GroupBox8.TabIndex = 7
        Me.GroupBox8.TabStop = False
        '
        'pnl_transTable
        '
        Me.pnl_transTable.Anchor =
CType((System.Windows.Forms.AnchorStyles.Top Or
System.Windows.Forms.AnchorStyles.Right),
System.Windows.Forms.AnchorStyles)
        Me.pnl_transTable.BackColor =
System.Drawing.Color.DarkSeaGreen
        Me.pnl_transTable.BorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D
        Me.pnl_transTable.Controls.Add(Me.dgv_m
ini)
        Me.pnl_transTable.Controls.Add(Me.Group
Box6)
```

```
        Me.pnl_transTable.Location = New
System.Drawing.Point(1011, 7)
        Me.pnl_transTable.Name =
"pnl_transTable"
        Me.pnl_transTable.Size = New
System.Drawing.Size(232, 229)
        Me.pnl_transTable.TabIndex = 0
        '
        'dgv_mini
        '
        Me.dgv_mini.AllowUserToAddRows = False
        Me.dgv_mini.AllowUserToDeleteRows =
False
        Me.dgv_mini.AllowUserToResizeColumns =
False
        Me.dgv_mini.AllowUserToResizeRows =
False
        DataGridViewCellStyle9.BackColor =
System.Drawing.Color.DarkSeaGreen
        Me.dgv_mini.AlternatingRowsDefaultCellS
tyle = DataGridViewCellStyle9
        Me.dgv_mini.AutoSizeColumnsMode =
System.Windows.Forms.DataGridViewAutoSizeColumn
sMode.Fill
        Me.dgv_mini.AutoSizeRowsMode =
System.Windows.Forms.DataGridViewAutoSizeRowsMo
de.AllCells
        Me.dgv_mini.BackgroundColor =
System.Drawing.Color.DarkSeaGreen
        Me.dgv_mini.ColumnHeadersBorderStyle =
System.Windows.Forms.DataGridViewHeaderBorderSt
yle.[Single]
        DataGridViewCellStyle10.Alignment =
System.Windows.Forms.DataGridViewContentAlignme
nt.MiddleRight
        DataGridViewCellStyle10.BackColor =
System.Drawing.SystemColors.Window
        DataGridViewCellStyle10.Font = New
System.Drawing.Font("Microsoft Sans Serif",
8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        DataGridViewCellStyle10.ForeColor =
System.Drawing.SystemColors.WindowText
        DataGridViewCellStyle10.SelectionBackCo
lor = System.Drawing.SystemColors.Highlight
        DataGridViewCellStyle10.SelectionForeCo
lor = System.Drawing.SystemColors.HighlightText
        DataGridViewCellStyle10.WrapMode =
System.Windows.Forms.DataGridViewTriState.
[True]
        Me.dgv_mini.ColumnHeadersDefaultCellSty
le = DataGridViewCellStyle10
        Me.dgv_mini.ColumnHeadersHeightSizeMode
=
System.Windows.Forms.DataGridViewColumnHeadersH
eightSizeMode.AutoSize
        DataGridViewCellStyle11.Alignment =
System.Windows.Forms.DataGridViewContentAlignme
nt.MiddleCenter
        DataGridViewCellStyle11.BackColor =
System.Drawing.Color.SpringGreen
        DataGridViewCellStyle11.Font = New
System.Drawing.Font("Microsoft Sans Serif",
8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        DataGridViewCellStyle11.ForeColor =
System.Drawing.SystemColors.ControlText
        DataGridViewCellStyle11.SelectionBackCo
lor = System.Drawing.SystemColors.Highlight
        DataGridViewCellStyle11.SelectionForeCo
lor = System.Drawing.SystemColors.HighlightText
        DataGridViewCellStyle11.WrapMode =
System.Windows.Forms.DataGridViewTriState.
[False]
        Me.dgv_mini.DefaultCellStyle =
DataGridViewCellStyle11
        Me.dgv_mini.EnableHeadersVisualStyles =
False
        Me.dgv_mini.GridColor =
System.Drawing.SystemColors.ActiveCaptionText
        Me.dgv_mini.Location = New
System.Drawing.Point(8, 21)
        Me.dgv_mini.MultiSelect = False
        Me.dgv_mini.Name = "dgv_mini"
        Me.dgv_mini.ReadOnly = True
        Me.dgv_mini.RowHeadersBorderStyle =
System.Windows.Forms.DataGridViewHeaderBorderSt
yle.[Single]
        DataGridViewCellStyle12.Alignment =
System.Windows.Forms.DataGridViewContentAlignme
nt.MiddleCenter
        DataGridViewCellStyle12.BackColor =
System.Drawing.SystemColors.Window
        DataGridViewCellStyle12.Font = New
System.Drawing.Font("Microsoft Sans Serif",
8.25!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        DataGridViewCellStyle12.ForeColor =
System.Drawing.SystemColors.WindowText
        DataGridViewCellStyle12.SelectionBackCo
lor = System.Drawing.SystemColors.Highlight
        DataGridViewCellStyle12.SelectionForeCo
lor = System.Drawing.SystemColors.HighlightText
        DataGridViewCellStyle12.WrapMode =
System.Windows.Forms.DataGridViewTriState.
[True]
        Me.dgv_mini.RowHeadersDefaultCellStyle
= DataGridViewCellStyle12
        Me.dgv_mini.RowHeadersWidthSizeMode =
System.Windows.Forms.DataGridViewRowHeadersWidt
hSizeMode.DisableResizing
        Me.dgv_mini.RowTemplate.DefaultCellStyl
e.BackColor =
System.Drawing.SystemColors.Window
        Me.dgv_mini.RowTemplate.ReadOnly = True
        Me.dgv_mini.ScrollBars =
System.Windows.Forms.ScrollBars.Vertical
        Me.dgv_mini.SelectionMode =
System.Windows.Forms.DataGridViewSelectionMode.
CellSelect
        Me.dgv_mini.ShowCellErrors = False
        Me.dgv_mini.ShowEditingIcon = False
        Me.dgv_mini.ShowRowErrors = False
        Me.dgv_mini.Size = New
System.Drawing.Size(209, 195)
        Me.dgv_mini.TabIndex = 1
        '
        'GroupBox6
        '
        Me.GroupBox6.Controls.Add(Me.Label1)
        Me.GroupBox6.Font = New
System.Drawing.Font("Arial", 9.0!,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.GroupBox6.Location = New
System.Drawing.Point(2, 3)
        Me.GroupBox6.Name = "GroupBox6"
        Me.GroupBox6.Size = New
System.Drawing.Size(221, 219)
        Me.GroupBox6.TabIndex = 2
        Me.GroupBox6.TabStop = False
        '
        'Label1
        '
        Me.Label1.Location = New
System.Drawing.Point(60, 0)
        Me.Label1.Name = "Label1"
        Me.Label1.Size = New
System.Drawing.Size(99, 13)
        Me.Label1.TabIndex = 0
        Me.Label1.Text = "Transition Table"
        '
        'pnl_statusAnimate
        '
        Me.pnl_statusAnimate.AutoSizeMode =
System.Windows.Forms.AutoSizeMode.GrowAndShrink
```

```
        Me.pnl_statusAnimate.BackColor =
System.Drawing.Color.DarkSeaGreen
        Me.pnl_statusAnimate.BorderStyle =
System.Windows.Forms.BorderStyle.Fixed3D
        Me.pnl_statusAnimate.Controls.Add(Me.Bu
tton2)
        Me.pnl_statusAnimate.Controls.Add(Me.Bu
tton1)
        Me.pnl_statusAnimate.Location = New
System.Drawing.Point(815, 510)
        Me.pnl_statusAnimate.Name =
"pnl_statusAnimate"
        Me.pnl_statusAnimate.Size = New
System.Drawing.Size(190, 89)
        Me.pnl_statusAnimate.TabIndex = 3
        '
        'Button2
        '
        Me.Button2.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.Button2.Location = New
System.Drawing.Point(42, 59)
        Me.Button2.Name = "Button2"
        Me.Button2.Size = New
System.Drawing.Size(102, 23)
        Me.Button2.TabIndex = 4
        Me.Button2.Text = "Clear Screen"
        Me.Button2.UseVisualStyleBackColor =
True
        '
        'Button1
        '
        Me.Button1.Font = New
System.Drawing.Font("Arial", 9.75!,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.Button1.ForeColor =
System.Drawing.Color.Chocolate
        Me.Button1.Location = New
System.Drawing.Point(3, 10)
        Me.Button1.Name = "Button1"
        Me.Button1.Size = New
System.Drawing.Size(180, 42)
        Me.Button1.TabIndex = 0
        Me.Button1.Text = "View Diagram"
        Me.Button1.UseVisualStyleBackColor =
True
        '
        'Panel2
        '
        Me.Panel2.BackColor =
System.Drawing.Color.DarkSeaGreen
        Me.Panel2.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle
        Me.Panel2.Controls.Add(Me.Label10)
        Me.Panel2.Controls.Add(Me.Label9)
        Me.Panel2.Controls.Add(Me.Label8)
        Me.Panel2.Controls.Add(Me.Label7)
        Me.Panel2.Controls.Add(Me.Label6)
        Me.Panel2.Controls.Add(Me.Label4)
        Me.Panel2.Controls.Add(Me.GroupBox4)
        Me.Panel2.Location = New
System.Drawing.Point(7, 509)
        Me.Panel2.Name = "Panel2"
        Me.Panel2.Size = New
System.Drawing.Size(175, 90)
        Me.Panel2.TabIndex = 0
        '
        'Label10
        '
        Me.Label10.BackColor =
System.Drawing.Color.Goldenrod
        Me.Label10.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle
        Me.Label10.Location = New
System.Drawing.Point(21, 62)
        Me.Label10.Name = "Label10"
        Me.Label10.Size = New
System.Drawing.Size(16, 10)
        Me.Label10.TabIndex = 6
        Me.Label10.Text = " "
        '
        'Label9
        '
        Me.Label9.BackColor =
System.Drawing.Color.RoyalBlue
        Me.Label9.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle
        Me.Label9.Location = New
System.Drawing.Point(21, 44)
        Me.Label9.Name = "Label9"
        Me.Label9.Size = New
System.Drawing.Size(16, 10)
        Me.Label9.TabIndex = 5
        Me.Label9.Text = " "
        '
        'Label8
        '
        Me.Label8.AutoSize = True
        Me.Label8.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.Label8.Location = New
System.Drawing.Point(43, 25)
        Me.Label8.Name = "Label8"
        Me.Label8.Size = New
System.Drawing.Size(58, 14)
        Me.Label8.TabIndex = 4
        Me.Label8.Text = "Start State"
        '
        'Label7
        '
        Me.Label7.AutoSize = True
        Me.Label7.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.Label7.Location = New
System.Drawing.Point(42, 62)
        Me.Label7.Name = "Label7"
        Me.Label7.Size = New
System.Drawing.Size(100, 14)
        Me.Label7.TabIndex = 3
        Me.Label7.Text = "StartAndFinal State"
        '
        'Label6
        '
        Me.Label6.AutoSize = True
        Me.Label6.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.Label6.Location = New
System.Drawing.Point(43, 44)
        Me.Label6.Name = "Label6"
        Me.Label6.Size = New
System.Drawing.Size(57, 14)
        Me.Label6.TabIndex = 2
        Me.Label6.Text = "Final State"
        '
        'Label4
        '
        Me.Label4.BackColor =
System.Drawing.Color.IndianRed
        Me.Label4.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle
        Me.Label4.Location = New
System.Drawing.Point(21, 26)
        Me.Label4.Name = "Label4"
        Me.Label4.Size = New
System.Drawing.Size(16, 10)
        Me.Label4.TabIndex = 0
        '
        'GroupBox4
        '
        Me.GroupBox4.AutoSizeMode =
System.Windows.Forms.AutoSizeMode.GrowAndShrink
        Me.GroupBox4.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.GroupBox4.Location = New
System.Drawing.Point(3, 1)
        Me.GroupBox4.Name = "GroupBox4"
        Me.GroupBox4.Size = New
System.Drawing.Size(167, 84)
        Me.GroupBox4.TabIndex = 7
        Me.GroupBox4.TabStop = False
        Me.GroupBox4.Text = "Legend"
        '
        'ToolStrip1
        '
        Me.ToolStrip1.AutoSize = False
        Me.ToolStrip1.BackColor =
System.Drawing.SystemColors.ActiveBorder
        Me.ToolStrip1.Dock =
System.Windows.Forms.DockStyle.None
        Me.ToolStrip1.GripStyle =
System.Windows.Forms.ToolStripGripStyle.Hidden
        Me.ToolStrip1.Items.AddRange(New
System.Windows.Forms.ToolStripItem()
{Me.ToolStripButton1, Me.ToolStripButton2})
        Me.ToolStrip1.Location = New
System.Drawing.Point(14, 36)
        Me.ToolStrip1.Name = "ToolStrip1"
        Me.ToolStrip1.Size = New
System.Drawing.Size(1249, 47)
        Me.ToolStrip1.TabIndex = 1
        Me.ToolStrip1.Text = "ToolStrip1"
        '
        'ToolStripButton1
        '
        Me.ToolStripButton1.AutoSize = False
        Me.ToolStripButton1.BackColor =
System.Drawing.Color.GhostWhite
        Me.ToolStripButton1.DisplayStyle =
System.Windows.Forms.ToolStripItemDisplayStyle.
Image
        Me.ToolStripButton1.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.ToolStripButton1.Image =
Global.SpecialProject.My.Resources.Resources.ne
w3
        Me.ToolStripButton1.ImageScaling =
System.Windows.Forms.ToolStripItemImageScaling.
None
        Me.ToolStripButton1.ImageTransparentCol
or = System.Drawing.Color.Magenta
        Me.ToolStripButton1.Name =
"ToolStripButton1"
        Me.ToolStripButton1.Size = New
System.Drawing.Size(53, 46)
        Me.ToolStripButton1.Text = "New"
        Me.ToolStripButton1.TextImageRelation =
System.Windows.Forms.TextImageRelation.Overlay
        '
        'ToolStripButton2
        '
        Me.ToolStripButton2.AutoSize = False
        Me.ToolStripButton2.BackColor =
System.Drawing.Color.GhostWhite
        Me.ToolStripButton2.DisplayStyle =
System.Windows.Forms.ToolStripItemDisplayStyle.
Image
        Me.ToolStripButton2.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.ToolStripButton2.Image =
Global.SpecialProject.My.Resources.Resources.co
nvert51
        Me.ToolStripButton2.ImageScaling =
System.Windows.Forms.ToolStripItemImageScaling.
None
        Me.ToolStripButton2.ImageTransparentCol
or = System.Drawing.Color.Magenta
        Me.ToolStripButton2.Name =
"ToolStripButton2"
        Me.ToolStripButton2.Size = New
System.Drawing.Size(53, 46)
        Me.ToolStripButton2.Text = "Convert"
        Me.ToolStripButton2.TextImageRelation =
System.Windows.Forms.TextImageRelation.Overlay
        '
        'status_REND
        '
        Me.status_REND.Location = New
System.Drawing.Point(0, 706)
        Me.status_REND.Name = "status_REND"
        Me.status_REND.Size = New
System.Drawing.Size(1276, 22)
        Me.status_REND.TabIndex = 2
        Me.status_REND.Text = "StatusStrip1"
        '
        'tmr_Animate
        '
        '
        'frm_REND
        '
        Me.AutoScaleDimensions = New
System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode =
System.Windows.Forms.AutoScaleMode.Font
        Me.AutoSize = True
        Me.AutoSizeMode =
System.Windows.Forms.AutoSizeMode.GrowAndShrink
        Me.AutoValidate =
System.Windows.Forms.AutoValidate.EnablePrevent
FocusChange
        Me.BackColor =
System.Drawing.SystemColors.Control
        Me.ClientSize = New
System.Drawing.Size(1276, 728)
        Me.Controls.Add(Me.ToolStrip1)
        Me.Controls.Add(Me.status_REND)
        Me.Controls.Add(Me.menuStrip_REND)
        Me.Controls.Add(Me.pnl_main)
        Me.Icon =
CType(resources.GetObject("$this.Icon"),
System.Drawing.Icon)
        Me.ImeMode =
System.Windows.Forms.ImeMode.[On]
        Me.MainMenuStrip = Me.menuStrip_REND
        Me.Name = "frm_REND"
        Me.StartPosition =
System.Windows.Forms.FormStartPosition.CenterSc
reen
        Me.Text = "frm_Main"
        Me.menuStrip_REND.ResumeLayout(False)
        Me.menuStrip_REND.PerformLayout()
        Me.pnl_main.ResumeLayout(False)
        Me.tabpages.ResumeLayout(False)
        Me.tab_transdiagram.ResumeLayout(False)
        Me.tab_transdiagram.PerformLayout()
        Me.pnl_diagram.ResumeLayout(False)
        Me.Panel1.ResumeLayout(False)
        Me.tab_transtable.ResumeLayout(False)
        Me.pnl_table.ResumeLayout(False)
        Me.Panel3.ResumeLayout(False)
        CType(Me.dgv_orig,
System.ComponentModel.ISupportInitialize).EndIn
it()
        Me.Panel4.ResumeLayout(False)
        Me.GroupBox2.ResumeLayout(False)
        CType(Me.DataGridView2,
System.ComponentModel.ISupportInitialize).EndIn
it()
        Me.pnl_pane.ResumeLayout(False)
        Me.Panel7.ResumeLayout(False)
        Me.GroupBox5.ResumeLayout(False)
        Me.GroupBox5.PerformLayout()
        Me.Panel6.ResumeLayout(False)
        Me.GroupBox3.ResumeLayout(False)
        Me.pnl_members.ResumeLayout(False)
        Me.GroupBox9.ResumeLayout(False)
        Me.GroupBox9.PerformLayout()
        Me.pnl_transFunc.ResumeLayout(False)
        Me.GroupBox7.ResumeLayout(False)
        Me.pnl_transTable.ResumeLayout(False)
        CType(Me.dgv_mini,
System.ComponentModel.ISupportInitialize).EndIn
it()
        Me.GroupBox6.ResumeLayout(False)
        Me.pnl_statusAnimate.ResumeLayout(False
)
        Me.Panel2.ResumeLayout(False)
        Me.Panel2.PerformLayout()
        Me.ToolStrip1.ResumeLayout(False)
        Me.ToolStrip1.PerformLayout()
        Me.ResumeLayout(False)
        Me.PerformLayout()

    End Sub
    Friend WithEvents menuStrip_REND As
System.Windows.Forms.MenuStrip
    Friend WithEvents tsm_File As
System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents tsm_New As
System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents tsm_Convert As
System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents tsm_ConvertToDFA As
System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents tsm_ConvertToRE As
System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents tsm_NewDFA As
System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents tsm_NewNFA As
System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents tsm_NewENFA As
System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents tsm_NewRE As
System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents tsm_Exit As
System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents tsm_View As
System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents tsm_viewTransitionTable
As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents
tsm_ViewTransitionFunction As
System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents tsm_About As
System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents pnl_main As
System.Windows.Forms.Panel
    Friend WithEvents pnl_transTable As
System.Windows.Forms.Panel
    Friend WithEvents pnl_transFunc As
System.Windows.Forms.Panel
    Friend WithEvents pnl_pane As
System.Windows.Forms.Panel
    Friend WithEvents pnl_statusAnimate As
System.Windows.Forms.Panel
    Friend WithEvents pnl_members As
System.Windows.Forms.Panel
    Friend WithEvents status_REND As
System.Windows.Forms.StatusStrip
    Friend WithEvents tmr_Animate As
System.Windows.Forms.Timer
    Friend WithEvents Button1 As
System.Windows.Forms.Button
    Friend WithEvents Panel2 As
System.Windows.Forms.Panel
    Friend WithEvents dgv_mini As
System.Windows.Forms.DataGridView
    Friend WithEvents Button2 As
System.Windows.Forms.Button
    Friend WithEvents RichTextBox1 As
System.Windows.Forms.RichTextBox
    Friend WithEvents tabpages As
System.Windows.Forms.TabControl
    Friend WithEvents tab_transdiagram As
System.Windows.Forms.TabPage
    Friend WithEvents pnl_diagram As
System.Windows.Forms.Panel
    Friend WithEvents Panel1 As
System.Windows.Forms.Panel
    Friend WithEvents pnl_Anim As
System.Windows.Forms.Panel
    Friend WithEvents tab_transtable As
System.Windows.Forms.TabPage
    Friend WithEvents pnl_table As
System.Windows.Forms.Panel
    Friend WithEvents Panel3 As
System.Windows.Forms.Panel
    Friend WithEvents dgv_orig As
System.Windows.Forms.DataGridView
    Friend WithEvents GroupBox1 As
System.Windows.Forms.GroupBox
    Friend WithEvents Panel4 As
System.Windows.Forms.Panel
    Friend WithEvents GroupBox2 As
System.Windows.Forms.GroupBox
    Friend WithEvents DataGridView2 As
System.Windows.Forms.DataGridView
    Friend WithEvents Panel5 As
System.Windows.Forms.Panel
    Friend WithEvents Label2 As
System.Windows.Forms.Label
    Friend WithEvents RichTextBox2 As
System.Windows.Forms.RichTextBox
    Friend WithEvents Label4 As
System.Windows.Forms.Label
    Friend WithEvents Label9 As
System.Windows.Forms.Label
    Friend WithEvents Label8 As
System.Windows.Forms.Label
    Friend WithEvents Label7 As
System.Windows.Forms.Label
    Friend WithEvents Label6 As
System.Windows.Forms.Label
    Friend WithEvents Label10 As
System.Windows.Forms.Label
    Friend WithEvents Panel6 As
System.Windows.Forms.Panel
    Friend WithEvents RichTextBox3 As
System.Windows.Forms.RichTextBox
    Friend WithEvents GroupBox3 As
System.Windows.Forms.GroupBox
    Friend WithEvents GroupBox4 As
System.Windows.Forms.GroupBox
    Friend WithEvents Panel7 As
System.Windows.Forms.Panel
    Friend WithEvents GroupBox5 As
System.Windows.Forms.GroupBox
    Friend WithEvents RichTextBox4 As
System.Windows.Forms.RichTextBox
    Friend WithEvents GroupBox6 As
System.Windows.Forms.GroupBox
    Friend WithEvents GroupBox7 As
System.Windows.Forms.GroupBox
    Friend WithEvents GroupBox8 As
System.Windows.Forms.GroupBox
    Friend WithEvents Label1 As
System.Windows.Forms.Label
    Friend WithEvents TextBox1 As
System.Windows.Forms.TextBox
    Friend WithEvents GroupBox9 As
System.Windows.Forms.GroupBox
    Friend WithEvents Button3 As
System.Windows.Forms.Button
    Friend WithEvents
HelpTopicsToolStripMenuItem As
System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents
AboutRENDToolStripMenuItem As
System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents ToolStrip1 As
System.Windows.Forms.ToolStrip
```

```
    Friend WithEvents ToolStripButton1 As
System.Windows.Forms.ToolStripButton
    Friend WithEvents ToolStripButton2 As
System.Windows.Forms.ToolStripButton
    Friend WithEvents LinkLabel1 As
System.Windows.Forms.LinkLabel
End Class
```

Frm_REND.vb

```
Imports Microsoft.DirectX
Imports Microsoft.DirectX.Direct3D
Imports Direct3D = Microsoft.DirectX.Direct3D

Public Class frm_REND

    Private device As Direct3D.Device
    Private angle As Single = 0.0
    Private Const max_X_left = -23
    Private Const max_X_right = 23
    Private Const max_Y_up = 8
    Private Const max_Y_down = -8
    Dim vertices As
CustomVertex.PositionColored()
    Dim StateVertices As
CustomVertex.PositionColored()
    Private mesh3DText As Mesh = Nothing 'Mesh
to draw 3d text.
    Private fontName As String = "Arial" 'Name
of the font you wish to use.
    Private FontSize As Integer = 10 'Size of
the font.
    Private textMaterial As Material 'Material
to color text.
    Private matFont As Matrix = Matrix.Identity
'Matrix to position and scale. text.
    Private currentFont As System.Drawing.Font
'Variable that stores the font.

    Private arrayTable As New ArrayList

    Private t As Threading.Thread = New
Threading.Thread(AddressOf paintDiagram)

    Private myEvents As New ArrayList
    Dim frmNumState As frm_numStates
    Dim pos As New StatePositions


    Private max_repulsive_dis As Integer = 40
    Private k As Integer = 13
    Private c As Double = 0.01
    Private max_vertex_move = 0.05
    Private time = 0
    Private go As Boolean = True
    Private init As Boolean = False
    Private tr As Integer = -10
    Private _index As Integer = 0

    Private _finalize As Boolean = False
    Private DONE As Boolean = False
    Dim present As PresentParameters
    Dim stateTemp As New State
    Dim numState As Integer = 0
    Dim arrayOfInputs As New ArrayList
    Dim view As Boolean = False
    Private autoTable As New DataGridView
    Dim aLineVectors(2) As
Microsoft.DirectX.Vector2
    Private aLine As Direct3D.Line
    Private p_type As String = Nothing
    Private draw As Boolean = True
    Private textPositions As New ArrayList
    Private p_re As RegEx
    Private myparent As frm_Main

    Public Sub New(ByVal p As frm_Main)
        InitializeComponent()
        myparent = p
    End Sub


    Public Sub Initialize()
        present = New PresentParameters
        present.Windowed = True 'we?ll draw on
a window

        present.SwapEffect = SwapEffect.Discard


        device = New Direct3D.Device(0,
DeviceType.Hardware, pnl_Anim,
CreateFlags.SoftwareVertexProcessing, present)
        device.Transform.Projection =
Matrix.PerspectiveFovLH(CSng(Math.PI / 4),
pnl_Anim.Width / pnl_Anim.Height, 1, 50) 'sets
field of view, aspect ratio, etc
        device.Transform.View =
Matrix.LookAtLH(New Vector3(0, 0, -30), New
Vector3(0, 0, 0), New Vector3(0, 1, 0))
'position and direction

        currentFont = New
System.Drawing.Font(fontName, FontSize)


        textMaterial = New Material
        textMaterial.AmbientColor = New
ColorValue(0, 0, 30)
        textMaterial.DiffuseColor = New
ColorValue(0, 0, 30)

        StateVertices = New
CustomVertex.PositionColored(1) {}

        aLine = New Direct3D.Line(device)

        aLineVectors(0).X = 10
        aLineVectors(0).Y = 10

        'Set the ending point of the line
        aLineVectors(1).X = 100
        aLineVectors(1).Y = 100

        aLineVectors(2).X = 200
        aLineVectors(2).Y = -100
    End Sub

    Public Sub setNumstate(ByVal x As Integer)
        numState = 0
        numState = x
    End Sub
    Public Sub setArrInputs(ByVal x As
ArrayList)
        arrayOfInputs = New ArrayList
        arrayOfInputs = x
    End Sub
    Private Sub frm_REND_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load

        Me.SetStyle(ControlStyles.UserPaint Or
ControlStyles.AllPaintingInWmPaint, True)

        Me.UpdateStyles()
        Me.Text = "REND"
        Initialize()

        tabpages.TabPages.Item(0).Text =
"Transition Diagram"

        t.Start()
        tmr_Animate = New Timer()
        tmr_Animate.Interval = 1
        tmr_Animate.Start()

    End Sub

    Private Sub paintDiagram()
```

```
        While go
            Try
                device.RenderState.Lighting =
False
                device.RenderState.CullMode =
Cull.None 'no triangle is culled
                device.Clear(ClearFlags.Target,
Color.Black, 1.0, 0)

                device.BeginScene() 'all
drawings after this line

                device.VertexFormat =
CustomVertex.PositionColored.Format
                device.Transform.World =
Matrix.Identity
                If draw Then
                    processEvents()
                End If
                DONE = True
                device.EndScene() 'all drawings
before this line
                Try
                    device.Present()
                Catch ex As Exception
                    device = Nothing
                End Try
            Catch e As Exception

            End Try



        End While
    End Sub

    Public Sub OnResetDevice(ByVal sender As
Object, ByVal e As EventArgs)

        If device Is Nothing And Me.Enabled
Then
            'Start the 3d Device
            device = Nothing
            Initialize()
        End If

    End Sub

    Public Sub processEvents()
        Try
            For i As Integer = 0 To
myEvents.Count - 1
                processCommand(CType(myEvents.I
tem(i), myEvent), i)
            Next
            If Not myEvents.Count = 0 Then
                If Not p_type = Nothing Then
                    For i As Integer = 0 To
myEvents.Count - 1
                        Dim e As myEvent =
myEvents(i)
                        If e.Command =
"DrawState" Then
                            Dim e1 As myEvent =
myEvents.Item(i)
                            Dim x As PointClass
= New PointClass(e1.m_State.x_Start1,
e1.m_State.y_Start1)
                            If Not
p_type.Equals("NFA-EE") Then
                                drawText(x, x,
e1.m_State.Ref, 0.8)
                            Else
                                drawText(x, x,
e1.m_State.Ref, 0.6)
                            End If

                        Else
                            Exit For
                        End If
                    Next
                    End If
                    For j As Integer = 0 To
textPositions.Count - 1
                        Dim arr As ArrayList =
textPositions(j)
                        Dim p1 As PointClass =
CType(arr(0), PointClass)
                        Dim p2 As PointClass =
CType(arr(1), PointClass)
                        Dim text As String =
CType(arr(2), String)
                        Dim size As Double =
CType(arr(3), Double)
                        drawText(p1, p2, text,
size)
                    Next

                End If


            Catch e As Exception

            End Try
        End Sub



        Public Sub processCommand(ByVal events As
myEvent, ByVal i As Integer)
            If events.Command = "DrawState" And
events.MustDo Then
                drawSphere(events.m_State)
            End If
            If events.Command.Equals("DrawArrow")
Then
                If i = numState Then
                    textPositions = New ArrayList
                End If
                Dim s As String = events.m_Arrow
                s = s.Replace("to", " ")
                Dim sA As Array = s.Split(" ")
                Dim index1 As Integer =
getIndex(sA(0))
                Dim index2 As Integer =
getIndex(sA(1))
                Dim e1 As myEvent =
myEvents.Item(index1)
                Dim e2 As myEvent =
myEvents.Item(index2)
                drawLine(e1.m_State, e2.m_State,
arrayOfInputs(i - numState), 20)
            End If
            If events.Command.Equals("DrawLine")
Then
                If i = numState Then
                    textPositions = New ArrayList
                End If
                Dim s As String = events.m_Arrow
                s = s.Replace("to", " ")
                Dim sA As Array = s.Split(" ")
                Dim index1 As Integer =
getIndex(sA(0))
                Dim index2 As Integer =
getIndex(sA(1))
                Dim e1 As myEvent =
myEvents.Item(index1)
                Dim e2 As myEvent =
myEvents.Item(index2)
                drawStraightLine(e1.m_State,
e2.m_State, arrayOfInputs(i - numState),
events.m_ArrowDir)
            End If
            If
events.Command.Equals("DrawObscureArrow") Then
                If i = numState Then
                    textPositions = New ArrayList
                End If
                Dim s As String = events.m_Arrow
                s = s.Replace("to", " ")
                Dim sA As Array = s.Split(" ")
            Dim index1 As Integer =
getIndex(sA(0))
            Dim index2 As Integer =
getIndex(sA(1))
            Dim e1 As myEvent =
myEvents.Item(index1)
            Dim e2 As myEvent =
myEvents.Item(index2)
            drawLine(e1.m_State, e2.m_State,
arrayOfInputs(i - numState), events.m_obs)
        End If
    End Sub

    Public Sub edit()
    End Sub




    Public Sub drawSphere(ByVal state1 As
State)
        Dim angle1 As Single = 0.0

        While angle1 <= 360
            state1.setVertices(state1.x_Start1
+ state1.radius * Math.Cos(CType(angle1 *
Math.PI / 180, Double)), _
                                  state1.y_Start1 +
state1.radius * Math.Sin(CType(angle1 * Math.PI
/ 180, Double)), _
                                  state1.z_Start1,
"(1)")
            'state.setAttribute()
            StateVertices(0).SetPosition(New
Vector3(state1.x_Start1, state1.y_Start1,
state1.z_Start1))
            StateVertices(0).Color =
Color.DarkSeaGreen.ToArgb
            StateVertices(1).SetPosition(New
Vector3(state1.x_Start2, state1.y_Start2,
state1.z_Start2))
            StateVertices(1).Color =
state1.color.ToArgb
            device.DrawUserPrimitives(Primitive
Type.LineList, 1, StateVertices)
            angle1 = angle1 + 1
        End While
    End Sub


    Public Sub Sequence(ByVal sender As Object,
ByVal e As EventArgs)
        init = True
    End Sub

    Public Sub drawStraightLine(ByVal state1 As
State, ByVal state2 As State, ByVal text As
String, ByVal dir As String)
        Dim p1 As PointClass = New
PointClass(state1.x_Start1, state1.y_Start1)
        Dim p2 As PointClass = New
PointClass(state2.x_Start1, state2.y_Start1)
        Dim d As Double = Math.Sqrt((p2.x -
p1.x) * (p2.x - p1.x) + (p2.y - p1.y) * (p2.y -
p1.y))
        Dim angle As Double =
Math.Acos((Math.Abs(p2.x - p1.x)) / d) * 180 /
Math.PI
        If p1.y > p2.y Then
            angle = -angle
        End If
        Dim x1 As Double = p1.x + state1.radius
* Math.Cos(angle * Math.PI / 180)
        Dim y1 As Double = p1.y + state1.radius
* Math.Sin(angle * Math.PI / 180)
        Dim x2 As Double = p2.x + state2.radius
* Math.Cos((angle + 180) * Math.PI / 180)
        Dim y2 As Double = p2.y + state2.radius
* Math.Sin((angle + 180) * Math.PI / 180)
        Dim vertices() As
CustomVertex.PositionColored = New
CustomVertex.PositionColored(1) {}
        vertices(0).Color =
Color.GreenYellow.ToArgb
        vertices(0).SetPosition(New Vector3(x1,
y1, 0))
        vertices(1).Color =
Color.GreenYellow.ToArgb
        vertices(1).SetPosition(New Vector3(x2,
y2, 0))
        Dim c1 As Double
        Dim c2 As Double
        Dim c3 As Double
        Dim c4 As Double
        If Not dir.Equals("left") Then
            c1 = p1.x + (d - 0.75) *
Math.Cos(angle * Math.PI / 180)
            c2 = p1.y + (d - 0.75) *
Math.Sin(angle * Math.PI / 180)
            c3 = p1.x + (d - 1.3) *
Math.Cos(angle * Math.PI / 180)
            c4 = p1.y + (d - 1.3) *
Math.Sin(angle * Math.PI / 180)
        Else
            c1 = p1.x + (state1.radius) *
Math.Cos(angle * Math.PI / 180)
            c2 = p1.y + (state1.radius) *
Math.Sin(angle * Math.PI / 180)
            c3 = p1.x + (state1.radius + 0.45)
* Math.Cos(angle * Math.PI / 180)
            c4 = p1.y + (state1.radius + 0.45)
* Math.Sin(angle * Math.PI / 180)
        End If

        Dim z1 As Double = p1.x + (d / 2) *
Math.Cos(angle * Math.PI / 180)
        Dim z3 As Double = p1.x + (d / 2 + 0.5)
* Math.Cos(angle * Math.PI / 180)
        Dim z2 As Double = 0
        Dim z4 As Double = 0
        If p1.y = p2.y Then
            z2 = p1.y + 0.4 + (d / 2) *
Math.Sin(angle * Math.PI / 180)
            z4 = p1.y + 0.4 + (d / 2 + 0.5) *
Math.Sin(angle * Math.PI / 180)
        Else
            z2 = p1.y + (d / 2) *
Math.Sin(angle * Math.PI / 180)
            z4 = p1.y + (d / 2 + 0.5) *
Math.Sin(angle * Math.PI / 180)
        End If

        Dim temp1 As PointClass = New
PointClass(c1, c2)
        Dim temp2 As PointClass = New
PointClass(c3, c4)
        DrawTriangle(temp2, temp1)
        Dim a As New ArrayList
        Dim tempp1 As PointClass = New
PointClass(z1, z2)
        Dim tempp2 As PointClass = New
PointClass(z3, z4)
        a.Add(tempp1)
        a.Add(tempp2)
        a.Add(text)
        a.Add(0.8)
        textPositions.Add(a)
        device.DrawUserPrimitives(PrimitiveType
.LineList, 1, vertices)
    End Sub
    Public Sub drawLine(ByVal state As State,
ByVal state2 As State, ByVal text As String,
ByVal obs As Integer)

        Dim arr As ArrayList
        If state.x_Start1 = state2.x_Start1 And
state.y_Start1 = state2.y_Start1 Then
            arr = getSixPoints(state, state2,
obs - 5)
        Else
```

```
            arr = getFourPoints(state, state2,
obs)
        End If
        arr = getCurve(arr)

        Dim vertices() As
CustomVertex.PositionColored = New
CustomVertex.PositionColored(arr.Count) {}
        For i As Integer = 0 To arr.Count - 1
            Dim x As PointClass = arr(i)
            vertices(i).SetPosition(New
Vector3(x.x, x.y, 0))
            vertices(i).Color =
Color.GreenYellow.ToArgb
            If i = arr.Count - 46 Then
                DrawTriangle(arr(i), arr(i +
2))
            End If
            If i = Math.Ceiling(arr.Count / 2)
Then
                Dim a As New ArrayList
                a.Add(arr(i))
                a.Add(arr(i + 1))
                a.Add(text)
                a.Add(0.8)
                textPositions.Add(a)
                'drawText(arr(i), arr(i + 1),
text)
            End If
        Next

        device.DrawUserPrimitives(PrimitiveType
.LineStrip, arr.Count - 1, vertices)
        'Dim aLineVEctors(arr.Count - 1) As
Microsoft.DirectX.Vector2
        'For i As Integer = 0 To arr.Count - 1
        'Dim a As PointClass = arr(i)
        'aLineVectors(i).X = 500 + a.x * 20
        'aLineVectors(i).Y = 250 - a.y * 20
        'Next
        'aLine.Width = 2
        'aLine.Draw(aLineVectors,
Color.Crimson.ToArgb)
    End Sub

    Public Sub drawText(ByVal p1 As PointClass,
ByVal p2 As PointClass, ByVal text As String,
ByVal size As Double)
        Dim d As Double = Math.Sqrt((p2.x -
p1.x) * (p2.x - p1.x) + (p2.y - p1.y) * (p2.y -
p1.y))
        Dim angle As Double =
Math.Acos((Math.Abs(p2.x - p1.x)) / d) * 180 /
Math.PI

        If p1.x < p2.x And p1.y > p2.y Then
            angle = 270 - angle
        End If
        If p1.x > p1.x And p1.y < p2.y Then
            angle = 90 - angle
        End If
        If p2.x > p1.x And p2.y > p1.y Then
            angle = angle + 90
        End If
        If p2.x < p1.x And p2.y < p1.y Then
            angle = -90 + angle
        End If
        If p1.x = p2.x Then
            If p1.y > p2.y Then
                angle = 180
            End If
            If p1.y < p2.y Then
                angle = 0
            End If
        End If

        If p1.y = p2.y Then
            If p1.x < p2.x Then
                angle = 90
            End If
            If p1.x > p2.x Then
                angle = 270
            End If
        End If

        If p1.x = p2.x And p1.y = p2.y Then
            angle = 90
        End If

        If angle - 90 < 0 Then
            angle = angle + 180
        End If
        Dim matRot1 As New Matrix
        matRot1.RotateYawPitchRoll(0, 0, (angle
- 90) * Math.PI / 180) 'XYZ
        'Set text mesh scale.
        Dim matScale1 As New Matrix
        If Size = 0.8 Then
            matScale1.Scale(0.8F, 0.8F, 0.1F)
'XYZ (Note: To change font depth change Z, a
higher value will result in more 3D look while
lower results in a flatter 2D look.)
        ElseIf size = 0.6 Then
            matScale1.Scale(0.6F, 0.6F, 0.1F)
'XYZ (Note: To change font depth change Z, a
higher value will result in more 3D look while
lower results in a flatter 2D look.)
        End If

        'Set text mesh position.
        Dim matPos1 As New Matrix
        If isSingleLabel(text) Then
            If size = 0.8 Then
                matPos1.Translate(p1.x - 0.25,
p1.y - 0.3, 0) 'XYZ
            Else
                matPos1.Translate(p1.x - 0.15,
p1.y - 0.3, 0) 'XYZ
            End If

        Else
            If size = 0.8 Then
                matPos1.Translate(p1.x - 0.5,
p1.y - 0.3, 0) 'XYZ
            Else
                matPos1.Translate(p1.x - 0.4,
p1.y - 0.3, 0) 'XYZ
            End If

        End If

        mesh3DText = Mesh.TextFromFont(device,
currentFont, text, 0.001F, 0.4F)
        device.Transform.World =
Matrix.Multiply(Matrix.Multiply(matRot1,
matScale1), matPos1)
        mesh3DText.DrawSubset(0) 'Render
Mesh/DrawText.
        device.Transform.World =
Matrix.Identity
    End Sub

    Public Function isSingleLabel(ByVal s As
String) As Boolean
        Dim a As New ArrayList
        a.Add("A") : a.Add("B") : a.Add("C") :
a.Add("D") : a.Add("E") : a.Add("F") :
a.Add("G") : a.Add("H") : a.Add("I") :
a.Add("J")
        a.Add("K") : a.Add("L") : a.Add("M") :
a.Add("N") : a.Add("O") : a.Add("P") :
a.Add("Q") : a.Add("R") : a.Add("S") :
a.Add("T")
        a.Add("U") : a.Add("V") : a.Add("W") :
a.Add("X") : a.Add("Y") : a.Add("Z")
        For i As Integer = 0 To a.Count - 1
            If a(i) = s Then
                Return True
            End If
        Next
        Return False
    End Function
    Public Sub DrawTriangle(ByVal p1 As
PointClass, ByVal p2 As PointClass)
        Dim d As Double = Math.Sqrt((p2.x -
p1.x) * (p2.x - p1.x) + (p2.y - p1.y) * (p2.y -
p1.y))
        Dim angle As Double =
Math.Acos((Math.Abs(p2.x - p1.x)) / d) * 180 /
Math.PI

        If p1.x < p2.x And p1.y > p2.y Then
            angle = 270 - angle
        End If
        If p1.x > p1.x And p1.y < p2.y Then
            angle = 90 - angle
        End If
        If p2.x > p1.x And p2.y > p1.y Then
            angle = angle + 90
        End If
        If p2.x < p1.x And p2.y < p1.y Then
            angle = -90 - angle
        End If
        If p1.x = p2.x Then
            If p1.y > p2.y Then
                angle = 180
            End If
            If p1.y < p2.y Then
                angle = 0
            End If
        End If

        If p1.y = p2.y Then
            If p1.x < p2.x Then
                angle = 90
            End If
            If p1.x > p2.x Then
                angle = 270
            End If
        End If


        Dim x1 As Double = p1.x + 0.3 *
Math.Cos(angle * Math.PI / 180)
        Dim y1 As Double = p1.y + 0.3 *
Math.Sin(angle * Math.PI / 180)
        Dim x2 As Double = p1.x + 0.3 *
Math.Cos((angle + 180) * Math.PI / 180)
        Dim y2 As Double = p1.y + 0.3 *
Math.Sin((angle + 180) * Math.PI / 180)

        Dim vertices() As
CustomVertex.PositionColored = New
CustomVertex.PositionColored(2) {}
        vertices(1).SetPosition(New
Vector3(p2.x, p2.y, 0))
        vertices(1).Color =
Color.YellowGreen.ToArgb
        vertices(0).SetPosition(New Vector3(x1,
y1, 0))
        vertices(0).Color =
Color.PaleGreen.ToArgb
        vertices(2).SetPosition(New Vector3(x2,
y2, 0))
        vertices(2).Color =
Color.YellowGreen.ToArgb
        device.DrawUserPrimitives(PrimitiveType
.TriangleStrip, 1, vertices)
    End Sub


    Public Function getCurve(ByVal arr As
ArrayList) As ArrayList
        Dim newArray As New ArrayList
        Dim tempArray As ArrayList = arr
        For i As Integer = 0 To 100
            newArray = New ArrayList
            newArray.Add(tempArray(0))
            For j As Integer = 0 To
tempArray.Count - 2
                Dim p1 As PointClass =
tempArray(j)
                    Dim p2 As PointClass =
tempArray(j + 1)
                    Dim mid As New PointClass((p1.x
+ p2.x) / 2, (p1.y + p2.y) / 2)
                    newArray.Add(mid)
                Next
                newArray.Add((tempArray(tempArray.C
ount - 1)))
                tempArray = New ArrayList
                tempArray = newArray
            Next
            Return tempArray
        End Function

        Public Function getFourPoints(ByVal state1
As State, ByVal state2 As State, ByVal _obs As
Integer) As ArrayList
            Dim a As New ArrayList
            Dim obs As Integer = _obs
            Dim d = Math.Sqrt((state2.x_Start1 -
state1.x_Start1) * (state2.x_Start1 -
state1.x_Start1) + (state2.y_Start1 -
state1.y_Start1) * (state2.y_Start1 -
state1.y_Start1))
            Dim angle As Double =
Math.Acos((Math.Abs(state2.x_Start1 -
state1.x_Start1)) / d) * 180 / Math.PI

            If state1.x_Start1 < state2.x_Start1
And state1.y_Start1 > state2.y_Start1 Then
                angle = 270 - angle
            End If
            If state1.x_Start1 > state2.x_Start1
And state1.y_Start1 < state2.y_Start1 Then
                angle = 90 - angle
            End If
            If state2.x_Start1 > state1.x_Start1
And state2.y_Start1 > state1.y_Start1 Then
                angle = angle + 90
            End If
            If state2.x_Start1 < state1.x_Start1
And state2.y_Start1 < state1.y_Start1 Then
                angle = -90 + angle
            End If
            If state1.x_Start1 = state2.x_Start1
Then
                If state1.y_Start1 >
state2.y_Start1 Then
                    angle = 180
                End If
                If state1.y_Start1 <
state2.y_Start1 Then
                    angle = 0
                End If
            End If

            If state1.y_Start1 = state2.y_Start1
Then
                If state1.x_Start1 <
state2.x_Start1 Then
                    angle = 90
                End If
                If state1.x_Start1 >
state2.x_Start1 Then
                    angle = 270
                End If
            End If


            Dim x1 As Double = state1.x_Start1 +
obs * Math.Cos(angle * Math.PI / 180)
            Dim x2 As Double = state2.x_Start1 +
obs * Math.Cos(angle * Math.PI / 180)
            Dim y1 As Double = state1.y_Start1 +
obs * Math.Sin(angle * Math.PI / 180)
            Dim y2 As Double = state2.y_Start1 +
obs * Math.Sin(angle * Math.PI / 180)

            Dim point1 As PointClass = New
PointClass(state1.x_Start1 + state1.radius *
Math.Cos(angle * Math.PI / 180),
state1.y_Start1 + state1.radius *
Math.Sin(angle * Math.PI / 180))
        Dim point4 As PointClass = New
PointClass(state2.x_Start1 + state2.radius *
Math.Cos(angle * Math.PI / 180),
state2.y_Start1 + state2.radius *
Math.Sin(angle * Math.PI / 180))
        Dim mid1 As PointClass = New
PointClass((x1 + x2) / 2, (y1 + y2) / 2)
        Dim point2 As PointClass = New
PointClass((mid1.x + point1.x) / 2, (mid1.y +
point1.y) / 2)
        Dim point3 As PointClass = New
PointClass((mid1.x + point4.x) / 2, (mid1.y +
point4.y) / 2)
        a.Add(point1)
        a.Add(point2)
        a.Add(point3)
        a.Add(point4)
        Return a
    End Function
    Public Function getSixPoints(ByVal state1
As State, ByVal state2 As State, ByVal _obs As
Integer) As ArrayList
        Dim a As New ArrayList
        Dim obs As Integer = _obs
        Dim height As Integer = 10
        Dim angle1 As Double = 0
        Dim angle2 As Double = 180


        Dim x1 As Double = state1.x_Start1 +
obs * Math.Cos(angle1 * Math.PI / 180)
        Dim x2 As Double = state2.x_Start1 +
obs * Math.Cos(angle2 * Math.PI / 180)
        Dim y1 As Double = state1.y_Start1 +
obs * Math.Sin(angle1 * Math.PI / 180)
        Dim y2 As Double = state2.y_Start1 +
obs * Math.Sin(angle2 * Math.PI / 180)
        Dim x3 As Double = x1
        Dim y3 As Double = y1 + height
        Dim x4 As Double = x2
        Dim y4 As Double = y2 + height

        Dim point1 As PointClass = New
PointClass(state1.x_Start1 + state1.radius *
Math.Cos(angle1 * Math.PI / 180),
state1.y_Start1 + state1.radius *
Math.Sin(angle1 * Math.PI / 180))
        Dim point6 As PointClass = New
PointClass(state2.x_Start1 + state2.radius *
Math.Cos(angle2 * Math.PI / 180),
state2.y_Start1 + state2.radius *
Math.Sin(angle2 * Math.PI / 180))
        Dim mid1 As PointClass = New
PointClass((x2 + x4) / 2, (y2 + y4) / 2)
        Dim mid2 As PointClass = New
PointClass((x4 + x3) / 2, (y4 + y3) / 2)
        Dim mid3 As PointClass = New
PointClass((x3 + x1) / 2, (y3 + y1) / 2)
        Dim point2 As PointClass = New
PointClass(mid1.x, mid1.y)
        Dim point3 As PointClass = New
PointClass((mid2.x + x4) / 2, (mid2.y + y4) /
2)
        Dim point4 As PointClass = New
PointClass((mid2.x + x3) / 2, (mid2.y + y3) /
2)
        Dim point5 As PointClass = New
PointClass(mid3.x, mid3.y)
        a.Add(point1)
        a.Add(point5)
        a.Add(point4)
        a.Add(point3)
        a.Add(point2)
        a.Add(point6)
        Return a
    End Function

    Public Sub LayoutME()
        If Not p_type Is Nothing Then
            If Not p_type.Equals("NFA-EE") Then
                LayoutIteration()
            End If
        End If


    End Sub

    Public Function getIndex(ByVal s As String)
As Integer
        For i As Integer = 0 To numState - 1
            Dim e As myEvent = myEvents(i)
            If e.Command = "DrawState" Then
                If
e.m_State.Ref.ToString.Equals(s) Then
                    Return i
                End If
            End If
        Next
        Return Nothing
    End Function
    Public Sub LayoutIteration()
        If myEvents.Count > 0 Then
            For i As Integer = 0 To numState -
1
                For j As Integer = i + 1 To
numState - 1
                    LayoutRepulsive(i, j)
                Next
            Next

            For i As Integer = numState To
myEvents.Count - 1
                Dim e As myEvent = myEvents(i)
                Dim s As String = e.m_Arrow
                s = s.Replace("to", " ")
                Dim sA As Array = s.Split(" ")
                Dim index1 As Integer =
getIndex(sA(0))
                Dim index2 As Integer =
getIndex(sA(1))
                LayoutAttractive(index1,
index2)
            Next



            For i As Integer = 0 To numState -
1
                Dim e As myEvent = myEvents(i)
                Dim state As State = e.m_State

                Dim xmove As Double = c *
state.force1
                Dim ymove As Double = c *
state.force2

                Dim max = max_vertex_move
                If xmove > max Then
                    xmove = max
                End If
                If xmove < -max Then
                    xmove = -max
                End If
                If ymove > max Then
                    ymove = max
                End If
                If ymove < -max Then
                    ymove = -max
                End If
                state.x_Start1 = state.x_Start1
+ xmove
                state.y_Start1 = state.y_Start1
+ ymove
                If state.x_Start1 < 0 And
state.x_Start1 < max_X_left Then
                    state.x_Start1 = max_X_left
                End If
                If state.x_Start1 > 0 And
state.x_Start1 > max_X_right Then
```

```
                    state.x_Start1 =
max_X_right
                End If
                If state.y_Start1 < 0 And
state.y_Start1 < max_Y_down Then
                    state.y_Start1 = max_Y_down
                End If
                If state.y_Start1 > 0 And
state.y_Start1 > max_Y_up Then
                    state.y_Start1 = max_Y_up
                End If

                state.force1 = 0
                state.force2 = 0
            Next
        End If
    End Sub

    Public Sub LayoutRepulsive(ByVal state_1 As
Integer, ByVal state_2 As Integer)
        Dim e1 As myEvent = myEvents(state_1)
        Dim e2 As myEvent = myEvents(state_2)
        Dim state1 As State = e1.m_State
        Dim state2 As State = e2.m_State
        Dim dy As Double = state2.y_Start1 -
state1.y_Start1
        Dim dx As Double = state2.x_Start1 -
state1.x_Start1

        Dim d2 As Double = dy * dy + dx * dx

        If d2 < 0.01 Then
            dx = Rnd(0.1) + 0.1
            dy = Rnd(0.1) + 0.1
            d2 = dy * dy + dx * dx
        End If

        Dim d = Math.Sqrt(d2)
        If d < max_repulsive_dis Then
            Dim repulsive_force = k * k / d
            state2.force1 = state2.force1 +
repulsive_force * dx / d
            state2.force2 = state2.force2 +
repulsive_force * dy / d
            state1.force1 = state1.force1 -
repulsive_force * dx / d
            state1.force2 = state1.force2 -
repulsive_force * dy / d
        End If
        e1.m_State = state1
        e2.m_State = state2
        myEvents(state_1) = e1
        myEvents(state_2) = e2

    End Sub
    Public Sub LayoutAttractive(ByVal state_1
As Integer, ByVal state_2 As Integer)
        Dim e1 As myEvent = myEvents(state_1)
        Dim e2 As myEvent = myEvents(state_2)
        Dim state1 As State = e1.m_State
        Dim state2 As State = e2.m_State
        Dim dy As Double = state2.y_Start1 -
state1.y_Start1
        Dim dx As Double = state2.x_Start1 -
state1.x_Start1
        Dim d2 As Double = dy * dy + dx * dx

        If d2 < 0.01 Then
            dx = Rnd(0.1) + 0.1
            dy = Rnd(0.1) + 0.1
            d2 = dy * dy + dx * dx
        End If
        Dim d = Math.Sqrt(d2)

        If d > max_repulsive_dis Then
            d = max_repulsive_dis
            d2 = d * d
        End If

        Dim attractive_force = (d2 - k * k) / k
        Dim weight As Double = 2
        attractive_force = attractive_force *
(Math.Log(weight) * 0.5 + 1)

        state2.force1 = state2.force1 -
attractive_force * dx / d
        state2.force2 = state2.force2 -
attractive_force * dy / d

        state1.force1 = state1.force1 +
attractive_force * dx / d
        state1.force2 = state1.force2 +
attractive_force * dy / d

        e1.m_State = state1
        e2.m_State = state2
        myEvents(state_1) = e1
        myEvents(state_2) = e2

    End Sub



    Public Sub setOriginalTable(ByVal table As
DataGridView)
        While dgv_orig.Columns.Count <> 0
            dgv_orig.Columns.Remove(dgv_orig.Co
lumns.Item(dgv_orig.Columns.Count -
1).Name.ToString)
        End While
        For i As Integer = 0 To
table.Columns.Count - 1
            dgv_orig.Columns.Add("column" &
i.ToString, table.Columns(i).HeaderText)
        Next
        For i As Integer = 0 To
table.Rows.Count - 1
            dgv_orig.Rows.Add()
            dgv_orig.Rows(i).HeaderCell.Value =
table.Rows.Item(i).HeaderCell.Value
        Next

        For k As Integer = 0 To
table.Columns.Count - 1
            For l As Integer = 0 To
table.Rows.Count - 1
                dgv_orig.Item(k, l).Value =
table.Item(k, l).Value
            Next
        Next
        RichTextBox1.Text = RichTextBox1.Text &
"Success!"
    End Sub

    Public Sub setMiniTable(ByVal table As
DataGridView)
        While dgv_mini.Columns.Count <> 0
            dgv_mini.Columns.Remove(dgv_mini.Co
lumns.Item(dgv_mini.Columns.Count -
1).Name.ToString)
        End While

        For i As Integer = 0 To
table.Columns.Count - 1
            dgv_mini.Columns.Add("column" &
i.ToString, table.Columns(i).HeaderText)
        Next
        For i As Integer = 0 To
table.Rows.Count - 1
            dgv_mini.Rows.Add()
            dgv_mini.Rows(i).HeaderCell.Value =
table.Rows.Item(i).HeaderCell.Value
        Next

        For k As Integer = 0 To
table.Columns.Count - 1
            For l As Integer = 0 To
table.Rows.Count - 1
                dgv_mini.Item(k, l).Value =
table.Item(k, l).Value
            Next
        Next
    End Sub

    Public Sub setConvertedTable(ByVal table As
DataGridView)
        While DataGridView2.Columns.Count <> 0
            Me.DataGridView2.Columns.Remove(Me.
DataGridView2.Columns.Item(DataGridView2.Column
s.Count - 1).Name.ToString)
        End While

        For i As Integer = 0 To
table.Columns.Count - 1
            Me.DataGridView2.Columns.Add("colum
n" & i.ToString, table.Columns(i).HeaderText)
        Next
        For i As Integer = 0 To
table.Rows.Count - 1
            Me.DataGridView2.Rows.Add()
            Me.DataGridView2.Rows(i).HeaderCell
.Value = table.Rows.Item(i).HeaderCell.Value
        Next

        For k As Integer = 0 To
table.Columns.Count - 1
            For l As Integer = 0 To
table.Rows.Count - 1
                Me.DataGridView2.Item(k,
l).Value = table.Item(k, l).Value
            Next
        Next
    End Sub



    Private Sub btn_TransTable_Click(ByVal
sender As System.Object, ByVal e As
System.EventArgs)



    End Sub





    Protected Overrides Sub OnPaint(ByVal e As
PaintEventArgs)
        'If Device has been lost or reset
        OnResetDevice(device, Nothing)

    End Sub

    Private Sub pnl_Anim_Paint(ByVal sender As
System.Object, ByVal e As
System.Windows.Forms.PaintEventArgs) Handles
pnl_Anim.Paint

    End Sub


    Public Sub setText(ByVal text As String)
        RichTextBox1.Text = RichTextBox1.Text &
vbCrLf & text
    End Sub

    Public Sub setRE(ByVal data As RegEx)
        p_re = data
    End Sub

    Public Sub setNew(ByVal com As String)
        If com.Equals("DFA") Then
            tsm_NewDFA_Click(Nothing, Nothing)
        ElseIf com.Equals("NFA") Then
            tsm_NewNFA_Click(Nothing, Nothing)
        ElseIf com.Equals("NFA-E") Then
            tsm_NewENFA_Click(Nothing, Nothing)
        ElseIf com.Equals("RE") Then
            tsm_NewRE_Click(Nothing, Nothing)
        End If
    End Sub
    Public Sub setTextRE(ByVal text As String)
        If text.Equals("Clear") Then
            RichTextBox2.Text = ""
        Else
            RichTextBox2.Text =
RichTextBox2.Text & text
        End If

    End Sub

    Public Sub setEvents(ByVal commands As
ArrayList)
        myEvents = New ArrayList
        myEvents = commands
    End Sub
    Public Sub setTable(ByVal table As
DataGridView)
        autoTable = table
    End Sub

    Private Sub tmr_Animate_Tick(ByVal sender
As System.Object, ByVal e As System.EventArgs)
Handles tmr_Animate.Tick
        If time < 100 Then
            If DONE = True And pnl_Anim.Visible
Then
                LayoutME()
                time = time + 1
                DONE = False
            End If
        Else
            Try
                t.Suspend()
            Catch ex As Exception

            End Try
        End If
    End Sub

    Private Sub frm_REND_Leave(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Leave
        Try
            t.Suspend()
        Catch ex As Exception

        End Try

        go = False
    End Sub

    Private Sub frm_REND_Enter(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Enter
        Try
            If
tabpages.TabPages(tabpages.SelectedIndex).Text.
Equals("Transition Diagram") Then
                t.Resume()
            End If

        Catch x As Exception
        End Try
        go = True
    End Sub

    Private Sub frm_REND_FormClosed(ByVal
sender As System.Object, ByVal e As
System.Windows.Forms.FormClosedEventArgs)
Handles MyBase.FormClosed
        myparent.RunningREND = False
        go = False
        Try
            t.Resume()
        Catch x As Exception
        End Try
        t.Abort()
    End Sub

    Private Sub tsm_NewDFA_Click(ByVal sender
As System.Object, ByVal e As System.EventArgs)
Handles tsm_NewDFA.Click
```

```
        If Not myEvents.Count = 0 Then
            myEvents.RemoveRange(0,
myEvents.Count)
        End If
        If Not textPositions.Count = 0 Then
            textPositions.RemoveRange(0,
textPositions.Count)
        End If
        frmNumState = New frm_numStates("DFA",
Me)
        frmNumState.Show(Me)
        Me.Enabled = False
        pnl_Anim.Visible = False
        p_type = "DFA"
        RichTextBox1.Text = "User inputs New
DFA..."
        GroupBox5.Text = "About Deterministic
Finite Automaton"
        RichTextBox4.Text = "Deterministic
refers to an instance wherein on each input
there is one and only one state which the
automaton can shift from its current state. A
transition diagram for a DFA A = (Q, Σ, δ, q0,
F) is a graph defined as follows:" & vbCrLf & "
a)       For each state in Q there is a node."
& _
        vbCrLf & " b)     For each state q in Q
and each symbol a in Σ, let δ(q, a)  = p. Then
the transition diagram has an arc from node q
to node p, labeled a. If there are several
input symbols that cause a transition from q to
p then the transition diagram can have one arc,
labeled by the list of these symbols. " & _
        vbCrLf & " c)     Start state is
labeled firebrick in color." & _
        vbCrLf & " d)     Nodes corresponding
to accepting states (those in F) are marked by
blue color. States not in F have single
circle."

    End Sub


    Private Sub tsm_NewNFA_Click(ByVal sender
As System.Object, ByVal e As System.EventArgs)
Handles tsm_NewNFA.Click
        If Not myEvents.Count = 0 Then
            myEvents.RemoveRange(0,
myEvents.Count)
        End If
        If Not textPositions.Count = 0 Then
            textPositions.RemoveRange(0,
textPositions.Count)
        End If
        frmNumState = New frm_numStates("NFA",
Me)
        frmNumState.Show(Me)
        Me.Enabled = False
        pnl_Anim.Visible = False
        p_type = "NFA"
        RichTextBox1.Text = "User inputs New
NFA..."
        GroupBox5.Text = "About Non-
Deterministic Finite Automaton"
        RichTextBox4.Text = "A 'non-
deterministic' finite automaton (NFA) has the
power to be in several states at once. " & _
                             "This ability is
often expressed as the ability to 'guess'
something about its input." & _
                             "The only
difference of an NFA to a DFA is its extended
transition function." & _
                             "As for the DFA's
we need to extend the transition function δ of
an NFA to a function δ^ that takes a state q
and a string of input symbols w, and returns
the set of states that the NFA is in if it is
starts in state q and processes the string w.
Thus, it is somewhat taking the union of the
states from the transition given an input a to
state q. Meaning, the language accepted by an
NFA is a string which has possibility to make
any sequence of choices of next state, while
reading the characters, and go from the start
state to at any accepting state." & _
                             "The fact that the
other choices using the input symbols lead to a
non accepting state, or die (do not lead to any
state at all)," & _
                             "does not prevent
the string to be accepted by the NFA as a whole
."
    End Sub

    Private Sub tsm_NewENFA_Click(ByVal sender
As System.Object, ByVal e As System.EventArgs)
Handles tsm_NewENFA.Click
        If Not myEvents.Count = 0 Then
            myEvents.RemoveRange(0,
myEvents.Count)
        End If
        If Not textPositions.Count = 0 Then
            textPositions.RemoveRange(0,
textPositions.Count)
        End If
        p_type = "NFA-E"
        frmNumState = New frm_numStates("NFA-
E", Me)
        frmNumState.Show()
        Me.Enabled = False
        pnl_Anim.Visible = False
        RichTextBox1.Text = "User inputs New ε-
NFA..."
        GroupBox5.Text = "About ε-Non-
Deterministic Finite Automaton"
        RichTextBox4.Text = "Another extension
of the finite automaton is a new feature that
allows a transition on ε, the empty string." &
_
                             "In effect, an NFA
is allowed to make a transition spontaneously
without receiving an input symbol. " & _
                             "Like the non-
determinism added, this new capability does not
expand the class of language that can be
accepted by finite automata, but it gives some
added 'programming convenience.'" & _
                             "We shall also see
later how NFA's with epsilon transition  which
we call ε-NFA's, are closely related to regular
expressions and useful in proving the
equivalence between the classes of languages
accepted by finite automata and by regular
expressions." & _
                             "We can represent ε
- NFA exactly as NFA except for one, the
transition function must include information
about transition on ε. We represent ε-NFA by A
= (Q, Σ, δ, q0, F) where all components are the
same as NFA, but δ is now a function that takes
as argument:" & _
                             vbCrLf & "1.   A
state in Q, and" & _
                             vbCrLf & "2.   A
member of Σ U {ε}, that is either an input
symbol, or the symbol ε. We may require that ε,
the empty string, cannot be a member of the
alphabet Σ, so no confusion results." & _
                             "We need to learn a
substantial definition called the ε-closure of
a state in computing ε-NFA's. Informally, ε-
closure ECLOSE(q) composed of the state itself
q and recursively all other states from q that
has an ε transition. "
    End Sub

    Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
```

```
        tabpages.SelectedTab =
tabpages.TabPages(0)
        pnl_Anim.Visible = True
        draw = True
        Try
            If
tabpages.TabPages(tabpages.SelectedIndex).Text.
Equals("Transition Diagram") Then
                t.Resume()
            End If

        Catch x As Exception
        End Try


        time = 0
    End Sub

    Private Sub dgv_mini_CellContentClick(ByVal
sender As System.Object, ByVal e As
System.Windows.Forms.DataGridViewCellEventArgs)
Handles dgv_mini.CellValueChanged
        dgv_mini.RowHeadersWidth = 50
        dgv_mini.Columns(0).Width = 50
    End Sub

    Private Sub tsm_ConvertToDFA_Click(ByVal
sender As System.Object, ByVal e As
System.EventArgs) Handles
tsm_ConvertToDFA.Click
        Dim m_DFA As DFA
        pnl_Anim.Visible = False
        Dim t As String = ""
        Try

            If Not p_type Is Nothing Then
                If p_type.Equals("NFA") Then
                    m_DFA = New
DFA(Me.dgv_orig, "convertNFA")
                    GroupBox5.Text =
"Converting from NFA to DFA"
                    RichTextBox1.AppendText(vbC
rLf & "User converts NFA to DFA...")
                    RichTextBox4.Text = "  The
language accepted by a DFA is exactly the same
language accepted by an NFA." & _
                                        "Th
us, we can conclude or it is not surprising
that there is equivalence of DFA and NFA." & _
                                        "Th
e conversion involves an important
'construction' called the subset construction
because it involves constructing all subsets of
the set of states of the NFA"
                    t = "NFA"
                ElseIf p_type.Equals("NFA-E")
Then
                    m_DFA = New
DFA(Me.dgv_orig, "convertNFA-E")
                    RichTextBox1.AppendText(vbC
rLf & "User converts ε-NFA to DFA...")
                    GroupBox5.Text =
"Converting from ε-NFA to DFA"
                    RichTextBox4.Text = "  The
language accepted by a DFA is exactly the same
language accepted by an ε-NFA." & _
                                        "Th
us, we can conclude or it is not surprising
that there is equivalence of DFA and ε-NFA." &
_
                                        "Th
e conversion involves an important
'construction' called the subset construction
(lke in NFA)because it involves constructing
all subsets of the set of states of the NFA." &
_
                                        "We
then apply the ECLOSE to get all transition
with empty input from the result"
                    t = "ε-NFA"
                End If
```

```
                Try
                    If p_type.Contains("NFA")
Then
                        setMiniTable(m_DFA.Tabl
e)
                        setConvertedTable(m_DFA
.ConvertTable)
                        setEvents(m_DFA.getComm
ands)
                        setNumstate(m_DFA.P_num
state)
                        setArrInputs(m_DFA.getI
nputs)
                        RichTextBox1.AppendText
("Success!")
                        p_type = "DFA"
                        setStatesEquivalence()
                        RichTextBox2.Text =
"Conversion Successful!" & vbCrLf & "Click View
Diagram to see the converted NFA."
                        createMEssage("Success"
, "Conversion Successful!", "Converting from "
& t & " to DFA...", "   Click view diagram
button to see the transition diagram or see
transition table for details.", Me)

                    Else
                        If p_type.Equals("DFA")
Then
                            RichTextBox1.Append
Text(vbCrLf & "Conversion Failed!" & vbCrLf &
"REND: Cannot convert DFA to itself!")
                            RichTextBox1.Append
Text(vbCrLf & "Suggestion: Convert to Regular
Expression")
                            createMEssage("Fail
ed", "Conversion Failed!", "Converting from DFA
to DFA...", "   Sorry but cannot convert DFA to
itself. Try converting it to Regular
Expression.", Me)
                        Else
                            RichTextBox1.Append
Text(vbCrLf & "Conversion Failed" & vbCrLf &
"REND: Cannot convert Regular Expression to
DFA!")
                            RichTextBox1.Append
Text(vbCrLf & "Suggestion: Convert to ε-NFA")
                            createMEssage("Fail
ed", "Conversion Failed!", "Converting from DFA
to DFA...", "   Sorry but cannot convert
Regular Expression to DFA. Try converting it to
DFA.", Me)
                        End If

                    End If

                Catch ex As Exception
                    RichTextBox1.AppendText(vbC
rLf & "Conversion Failed!")
                    createMEssage("Failed",
"Conversion Failed!", "Converting to DFA...", "
Sorry!Cannot process conversion.", Me)
                End Try
            Else
                RichTextBox1.AppendText(vbCrLf
& "Conversion Failed!..." & vbCrLf & "REND:
Please enter a representation first!")
                createMEssage("Failed",
"Conversion Failed!", "Converting from null to
DFA...", "   Sorry but you have not entered a
representation.", Me)
            End If

        Catch ex As Exception
            RichTextBox1.AppendText(vbCrLf &
"Conversion Failed!..." & vbCrLf & "REND:
Please enter a representation first!")
            createMEssage("Failed", "Conversion
Failed!", "Converting from null to DFA...", "
Sorry but you have not entered a
representation.", Me)
        End Try



    End Sub

    Public Sub createMEssage(ByVal title As
String, ByVal text As String, ByVal type As
String, ByVal message As String, ByVal p As
Form)
        Me.Enabled = False
        Dim m As Message = New Message(title,
text, type, message, Me)
        m.Show()
    End Sub


    Public Sub setStatesEquivalence()
        Dim statesEquivalence As String = ""
        For i As Integer = 0 To
dgv_mini.Rows.Count - 2
            statesEquivalence =
statesEquivalence &
dgv_mini.Rows(i).HeaderCell.Value & "-" &
DataGridView2.Rows(i).HeaderCell.Value & vbCrLf
        Next
        RichTextBox3.Text = statesEquivalence
    End Sub


    Private Sub
DataGridView2_CellContentClick(ByVal sender As
System.Object, ByVal e As
System.Windows.Forms.DataGridViewCellEventArgs)
Handles DataGridView2.CellContentClick

    End Sub

    Private Sub tsm_ConvertToNFA_Click(ByVal
sender As System.Object, ByVal e As
System.EventArgs)

    End Sub

    Private Sub Button2_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button2.Click
        draw = False
        pnl_Anim.Visible = False
        Try
            t.Suspend()
        Catch ex As Exception

        End Try

    End Sub

    Private Sub pnl_Anim_Enter(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles pnl_Anim.Enter
        Try
            If
tabpages.TabPages(tabpages.SelectedIndex).Text.
Equals("Transition Diagram") Then
                t.Resume()
                pnl_Anim.Visible = True
                draw = True
            End If

        Catch x As Exception
        End Try
        go = True
    End Sub

    Private Sub pnl_Anim_Leave(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles pnl_Anim.Leave
        Try
            t.Suspend()
            draw = False
        Catch ex As Exception

        End Try

        go = False
    End Sub

    Private Sub tsm_ConvertToRE_Click(ByVal
sender As System.Object, ByVal e As
System.EventArgs) Handles tsm_ConvertToRE.Click
        Try
            pnl_Anim.Visible = False
            If Not dgv_mini Is Nothing And
p_type.Equals("DFA") Then
                RichTextBox1.AppendText(vbCrLf
& "User converts DFA to Regular Expression...")
                Me.Enabled = False
                Dim re As New
Regular_Expression(dgv_mini, Me)
                GroupBox5.Text = "Converting
from DFA to Regular Expression"
                RichTextBox4.Text = "Regular
Expression also define the language of an
automaton. The equivalence can be shown by
elimination method:"
            Else
                RichTextBox1.AppendText(vbCrLf
& "User tries to convert NFA to Regular
Expression...Failed")
                RichTextBox1.AppendText(vbCrLf
& "REND: Please convert NFA to DFA first before
converting to RE.")
                createMEssage("Failed",
"Conversion Failed!", "Converting from NFA to
Regular Expression...", "   Please convert NFA
to DFA first.", Me)
            End If
        Catch ex As Exception
            RichTextBox1.AppendText(vbCrLf &
"REND: Please enter a representation first!")
            createMEssage("Failed", "Conversion
Failed!", "Cannot process conversion...", "
Sorry. Cannot continue conversion.", Me)
        End Try

    End Sub



    Private Sub Label5_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)

    End Sub

    Private Sub
tabpages_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles tabpages.SelectedIndexChanged
        If Not
tabpages.TabPages(tabpages.SelectedIndex).Text.
Equals("Transition Diagram") Then
            Try
                t.Suspend()
                pnl_Anim.Visible = False
                draw = False
            Catch ex As Exception

            End Try

        Else
            Try
                t.Resume()
                pnl_Anim.Visible = True
                draw = True
            Catch ex As Exception

            End Try

        End If
    End Sub
    Private Sub tsm_NewRE_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles tsm_NewRE.Click
        If Not myEvents.Count = 0 Then
            myEvents.RemoveRange(0,
myEvents.Count)
        End If
        If Not textPositions.Count = 0 Then
            textPositions.RemoveRange(0,
textPositions.Count)
        End If
        While dgv_mini.Rows.Count <> 0
            dgv_mini.Rows.Remove(dgv_mini.Rows(
dgv_mini.Rows.Count - 1))
        End While
        While dgv_mini.Columns.Count <> 0
            dgv_mini.Columns.Remove(dgv_mini.Co
lumns(dgv_mini.Columns.Count - 1))
        End While
        While dgv_orig.Rows.Count <> 0
            dgv_orig.Rows.Remove(dgv_orig.Rows(
dgv_orig.Rows.Count - 1))
        End While
        While dgv_orig.Columns.Count <> 0
            dgv_orig.Columns.Remove(dgv_orig.Co
lumns(dgv_orig.Columns.Count - 1))
        End While
        While DataGridView2.Rows.Count <> 0
            DataGridView2.Rows.Remove(DataGridV
iew2.Rows(DataGridView2.Rows.Count - 1))
        End While
        While DataGridView2.Columns.Count <> 0
            DataGridView2.Columns.Remove(DataGr
idView2.Columns(DataGridView2.Columns.Count -
1))
        End While
        p_type = "NFA-EE"
        Me.Enabled = False
        pnl_Anim.Visible = False
        Dim frmRE As New frm_RE(Me)
        frmRE.Show()
        RichTextBox1.Text = "User inputs New
Regular Expression...Success!"
        GroupBox5.Text = "About Regular
Expression"
        RichTextBox4.Text = "Another notation
that can describe a regular language is the
algebraic description called 'regular
expression'." & _
                            "We shall see that
the regular expression defines exactly the same
language as the various forms of automata
describe." & _
                            "However, regular
expressions offer something that is not visible
in automata, the declarative way to express the
strings we want to accept." & _
                            "It somewhat serve
as the input language for many systems that
process a string."
    End Sub


    Private Sub
tsm_viewTransitionTable_Click_1(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles tsm_viewTransitionTable.Click
        tabpages.SelectedTab =
tabpages.TabPages(1)
    End Sub

    Private Sub
tsm_ViewTransitionFunction_Click_1(ByVal sender
As System.Object, ByVal e As System.EventArgs)
Handles tsm_ViewTransitionFunction.Click
        tabpages.SelectedTab =
tabpages.TabPages(0)
    End Sub
```

```
    Private Sub LinkLabel1_LinkClicked(ByVal
sender As System.Object, ByVal e As
System.Windows.Forms.LinkLabelLinkClickedEventA
rgs) Handles LinkLabel1.LinkClicked
        Dim r As New frm_Reference
        r.Show()
    End Sub

    Private Sub Button3_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button3.Click
        If TextBox1.Text.Equals("") Then
            createMEssage("No input", "Failed",
"Testing a null string is not valid.", "
Please enter a string consisiting of 0's and
1's.", Me)
        Else
            If Not p_type Is Nothing Then
                If p_type.Equals("DFA") Or
p_type.Equals("NFA") Or p_type.Equals("NFA-E")
Then
                    checkMembership(TextBox1.Te
xt)
                End If

                If p_type.Equals("NFA-EE") Then
                    checkMembershipInRE(TextBox
1.Text)
                End If
            End If
        End If

    End Sub
    Public Sub checkMembershipInRE(ByVal data
As String)
        Dim z As Array = data.ToCharArray
        Dim current As New ArrayList
        current.Add(0)
        For i As Integer = 0 To z.Length - 1
            current = getNextStates(current,
z(i).ToString)
        Next
        If current.Count > 0 Then
            If Not current(current.Count -
1).Equals(4) Then
                If
checkifallempty(current(current.Count - 1))
Then
                    createMEssage("Member!",
"(Success)It is a member", "Testing
Membership...", "The string " & TextBox1.Text &
" is a member of the language.", Me)
                Else
                    createMEssage("Not
Member!", "It is not a member", "Testing
Membership...", "The string " & TextBox1.Text &
" is not a member of the language.", Me)
                End If
            Else
                createMEssage("Member!",
"(Success)It is a member", "Testing
Membership...", "The string " & TextBox1.Text &
" is a member of the language.", Me)
            End If
        Else
            createMEssage("Not Member!", "It is
not a member", "Testing Membership...", "The
string " & TextBox1.Text & " is not a member of
the language.", Me)
        End If
    End Sub

    Public Function checkifallempty(ByVal c As
Integer) As Boolean
        Select Case c
            Case Is = 1
                If
p_re.secondTerm.Contains("*") And
p_re.thirdTerm.Contains("*") And
p_re.fourthTerm.Contains("*") Then
                    Return True
                End If
            Case Is = 2
                If p_re.thirdTerm.Contains("*")
And p_re.fourthTerm.Contains("*") Then
                    Return True
                End If
            Case Is = 3
                If
p_re.fourthTerm.Contains("*") Then
                    Return True
                End If
        End Select
        Return False
    End Function

    Public Function getNextStates(ByVal c As
ArrayList, ByVal data As String) As ArrayList
        Dim arr As New ArrayList
        For i As Integer = 0 To c.Count - 1
            Dim a As ArrayList =
getIndeces(c(i), data)
            For j As Integer = 0 To a.Count - 1
                If Not arr.Contains(a(j)) Then
                    arr.Add(a(j))
                End If
            Next
        Next
        Return arr
    End Function

    Public Function getIndeces(ByVal c As
Integer, ByVal data As String) As ArrayList
        If c = 0 Then
            Return getNext(p_re.firstTerm, c,
data)
        ElseIf c = 1 Then
            Return getNext(p_re.secondTerm, c,
data)
        ElseIf c = 2 Then
            Return getNext(p_re.thirdTerm, c,
data)
        ElseIf c = 3 Then
            Return getNext(p_re.fourthTerm, c,
data)
        ElseIf c = 4 Then
            Return getNext(p_re.fourthTerm, c,
data)
        End If
    End Function
    Public Function getNext(ByVal com As
String, ByVal term As Integer, ByVal data As
String)
        Dim arr As New ArrayList
        Select Case com
            Case Is = "0"
                If data.Equals("0") Then
                    If Not term = 4 Then
                        arr.Add(term + 1)
                    End If
                End If
            Case Is = "1"
                If data.Equals("1") Then
                    If Not term = 4 Then
                        arr.Add(term + 1)
                    End If
                End If
            Case Is = "0*"
                If data.Equals("0") Then
                    arr.Add(term)
                    If Not term = 4 Then
                        arr.Add(term + 1)
                    End If
                End If
            Case Is = "1*"
                If data.Equals("1") Then
                    arr.Add(term)
                    If Not term = 4 Then
                        arr.Add(term + 1)
                    End If
                End If
            Case Is = "(0+1)"
                If data.Equals("0") Or
data.Equals("1") Then
                    If Not term = 4 Then
                        arr.Add(term + 1)
                    End If
                End If
            Case Is = "(0+1)*"
                If data.Equals("1") Or
data.Equals("0") Then
                    arr.Add(term)
                    If Not term = 4 Then
                        arr.Add(term + 1)
                    End If
                End If
        End Select
        Return arr
    End Function
    Public Sub checkMembership(ByVal text As
String)
        If dgv_mini.Rows.Count = 0 Then
            MsgBox("no DFA data")
        Else
            Dim z As Array = text.ToCharArray
            Dim current As ArrayList =
findStart()
            Dim finals As ArrayList =
findFinal()
            For i As Integer = 0 To z.Length -
1
                current =
getCurrentStates(current, z(i).ToString)
            Next
            For i As Integer = 0 To
current.Count - 1
                If finals.Contains(current(i))
Then
                    createMEssage("Member!",
"(Success)It is a member", "Testing
Membership...", "The string " & TextBox1.Text &
" is a member of the language.", Me)
                    Exit Sub
                End If
            Next
            createMEssage("Not Member!", "It is
not a member", "Testing Membership...", "The
string " & TextBox1.Text & " is not a member of
the language.", Me)
        End If
    End Sub

    Private Sub TextBox1_TextChanged(ByVal
sender As System.Object, ByVal e As
System.EventArgs) Handles TextBox1.TextChanged
        Dim z As Array =
TextBox1.Text.ToCharArray
        For i As Integer = 0 To
TextBox1.Text.Length - 1
            If Not z(i).ToString.Equals("1")
And Not z(i).ToString.Equals("0") Then
                TextBox1.Text =
TextBox1.Text.Replace(z(i).ToString, "")
                TextBox1.SelectionStart =
TextBox1.Text.Length
                TextBox1.SelectionLength = 0
                TextBox1.ScrollToCaret()
            End If
        Next
    End Sub

    Public Function findStart() As ArrayList
        Dim arr As New ArrayList
        For i As Integer = 0 To
dgv_mini.Rows.Count - 2
            If dgv_mini.Item(0,
i).Value.Contains("Start") Then
                arr.Add(i)
            End If
        Next
        Return arr
    End Function
    Public Function findFinal() As ArrayList
```

```
        Dim arr As New ArrayList
        Dim count As Integer = 0
        If p_type.Equals("NFA") Or
p_type.Equals("NFA-E") Then
            count = 1
        ElseIf p_type.Equals("DFA") Then
            count = 2
        End If
        For i As Integer = 0 To
dgv_mini.Rows.Count - count
            If dgv_mini.Item(0,
i).Value.Contains("Final") Then
                arr.Add(i)
            End If
        Next
        Return arr
    End Function
    Public Function getCurrentStates(ByVal
states As ArrayList, ByVal data As String) As
ArrayList
        Dim arr As New ArrayList
        For i As Integer = 0 To states.Count -
1
            For j As Integer = 0 To
dgv_mini.Columns.Count - 1
                If
dgv_mini.Columns(j).HeaderText.Equals(data)
Then
                    If Not dgv_mini.Item(j,
states(i)).Value.ToString.Equals("ø") Then
                        If Not
p_type.Equals("NFA-E") Then
                            arr.Add(dgv_mini.It
em(j, states(i)).Value.ToString)
                        Else
                            arr.Add(getEclose(d
gv_mini.Item(j, states(i)).Value.ToString))
                        End If

                    End If
                End If
            Next
        Next
        Dim a As ArrayList = getValue(arr)
        Return a
    End Function

    Public Function getRowNumber(ByVal data As
String) As Integer
        Dim endCount As Integer = 0
        If p_type.Equals("DFA") Then
            endCount = dgv_mini.Rows.Count - 2
        Else
            endCount = dgv_mini.Rows.Count - 1
        End If
        For i As Integer = 0 To endCount
            If
dgv_mini.Rows(i).HeaderCell.Value.Equals(data)
Then
                Return i
            End If
        Next
    End Function

    Public Function getEclose(ByVal data As
String) As String
        Dim value As String = ""
        If data.Contains("{") Then
            value = "{"
            Dim d As String = data.Replace("{",
"")
            d = d.Replace("}", "")
            d = d.Replace(",", " ")
            Dim z As Array = d.Split(" ")
            For i As Integer = 0 To z.Length -
1
                Dim c As String =
dgv_mini.Item(1,
getRowNumber(z(i).ToString)).Value.ToString
                If c.Contains("{") Then
                    Dim x As String =
c.Replace("{", "")
                    x = x.Replace("}", "")
                    x = x.Replace(",", " ")
                    Dim y As Array = x.Split("
")
                    For j As Integer = 0 To
y.Length - 1
                        value = value &
y(j).ToString & ","
                    Next
                Else
                    value = value & c & ","
                End If
            Next
            value = value.Remove(value.Length -
1)
            value = value & "}"
        Else
            value = dgv_mini.Item(1,
getRowNumber(data)).Value.ToString
        End If
        Return value
    End Function
    Public Function getValue(ByVal a As
ArrayList) As ArrayList
        Dim value As New ArrayList
        Dim z As ArrayList =
getIndividualValue(a)
        For i As Integer = 0 To z.Count - 1
            value.Add(getRowNumber(z(i)))
        Next
        Return value
    End Function

    Public Function getIndividualValue(ByVal a
As ArrayList) As ArrayList
        Dim arr As New ArrayList
        For i As Integer = 0 To a.Count - 1
            If a(i).ToString.Contains("{") Then
                Dim str As String = a(i)
                str = str.Replace("{", "")
                str = str.Replace("}", "")
                str = str.Replace(",", " ")
                Dim z As Array = str.Split(" ")
                For j As Integer = 0 To
z.Length - 1
                    If Not
arr.Contains(z(j).ToString) Then
                        arr.Add(z(j).ToString)
                    End If
                Next
            Else
                If Not
arr.Contains(a(i).ToString) Then
                    arr.Add(a(i).ToString)
                End If
            End If
        Next
        Return arr
    End Function

    Private Sub tsm_New_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles tsm_New.Click

    End Sub

    Private Sub ToolStripButton2_Click(ByVal
sender As System.Object, ByVal e As
System.EventArgs) Handles
ToolStripButton2.Click
        If p_type Is Nothing Then
            createMEssage("Error", "Conversion
Failed", "Conversion from null...", "    Please
enter a representation first.", Me)
        ElseIf p_type.Equals("NFA") Or
p_type.Equals("NFA-E") Then
            tsm_ConvertToDFA_Click(Nothing,
Nothing)
        ElseIf p_type.Equals("DFA") Then
```

```
            tsm_ConvertToRE_Click(Nothing,
Nothing)
        ElseIf p_type.Equals("NFA-EE") Then
            createMEssage("Success", "Auto-
Conversion Successful!", "Conversion from RE to
ε-NFA", "    Click view diagram to see
transition diagram.", Me)
        End If
    End Sub

    Private Sub ToolStripButton1_Click(ByVal
sender As System.Object, ByVal e As
System.EventArgs) Handles
ToolStripButton1.Click
        Dim n As New NewForm(Me)
        n.Show()
        Me.Enabled = False
    End Sub

    Private Sub
HelpTopicsToolStripMenuItem_Click(ByVal sender
As System.Object, ByVal e As System.EventArgs)
Handles HelpTopicsToolStripMenuItem.Click
        Dim r As New frm_Reference
        r.Show()
    End Sub

    Private Sub
AboutRENDToolStripMenuItem_Click(ByVal sender
As System.Object, ByVal e As System.EventArgs)
Handles AboutRENDToolStripMenuItem.Click
        Dim about As About = New About()
        about.Show()
    End Sub
End Class
```

Frm_Numstate.designer.vb

```
<Global.Microsoft.VisualBasic.CompilerServices.
DesignerGenerated()> _
Partial Class frm_numStates
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the
component list.
    <System.Diagnostics.DebuggerNonUserCode()>
_
    Protected Overrides Sub Dispose(ByVal
disposing As Boolean)
        Try
            If disposing AndAlso components
IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As
System.ComponentModel.IContainer

    'NOTE: The following procedure is required
by the Windows Form Designer
    'It can be modified using the Windows Form
Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()>
_
    Private Sub InitializeComponent()
        Dim resources As
System.ComponentModel.ComponentResourceManager
= New
System.ComponentModel.ComponentResourceManager(
GetType(frm_numStates))
        Me.lbl_numStates = New
System.Windows.Forms.Label
        Me.cmb_numStates = New
System.Windows.Forms.ComboBox
        Me.Panel1 = New
System.Windows.Forms.Panel
        Me.Panel2 = New
System.Windows.Forms.Panel
        Me.Button2 = New
System.Windows.Forms.Button
        Me.Button1 = New
System.Windows.Forms.Button
        Me.Panel3 = New
System.Windows.Forms.Panel
        Me.Panel1.SuspendLayout()
        Me.Panel2.SuspendLayout()
        Me.Panel3.SuspendLayout()
        Me.SuspendLayout()
        '
        'lbl_numStates
        '
        Me.lbl_numStates.AutoSize = True
        Me.lbl_numStates.BackColor =
System.Drawing.Color.DarkSeaGreen
        Me.lbl_numStates.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.lbl_numStates.Location = New
System.Drawing.Point(13, 24)
        Me.lbl_numStates.Name = "lbl_numStates"
        Me.lbl_numStates.Size = New
System.Drawing.Size(106, 14)
        Me.lbl_numStates.TabIndex = 0
        Me.lbl_numStates.Text = "Number of
states:"
        '
        'cmb_numStates
        '
        Me.cmb_numStates.DisplayMember =
"(none)"
        Me.cmb_numStates.DropDownStyle =
System.Windows.Forms.ComboBoxStyle.DropDownList
        Me.cmb_numStates.FormattingEnabled =
True
        Me.cmb_numStates.Items.AddRange(New
Object() {"1", "2", "3", "4", "5", "6", "7",
"8", "9", "10"})
        Me.cmb_numStates.Location = New
System.Drawing.Point(134, 21)
        Me.cmb_numStates.Name = "cmb_numStates"
        Me.cmb_numStates.Size = New
System.Drawing.Size(63, 22)
        Me.cmb_numStates.TabIndex = 1
        '
        'Panel1
        '
        Me.Panel1.BackColor =
System.Drawing.Color.DarkSeaGreen
        Me.Panel1.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle
        Me.Panel1.Controls.Add(Me.cmb_numStates
)
        Me.Panel1.Controls.Add(Me.lbl_numStates
)
        Me.Panel1.Location = New
System.Drawing.Point(1, 1)
        Me.Panel1.Name = "Panel1"
        Me.Panel1.Size = New
System.Drawing.Size(218, 65)
        Me.Panel1.TabIndex = 16
        '
        'Panel2
        '
        Me.Panel2.BackColor =
System.Drawing.Color.DarkSeaGreen
        Me.Panel2.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle
        Me.Panel2.Controls.Add(Me.Button2)
        Me.Panel2.Controls.Add(Me.Button1)
        Me.Panel2.Controls.Add(Me.Panel3)
        Me.Panel2.Location = New
System.Drawing.Point(1, 3)
        Me.Panel2.Name = "Panel2"
        Me.Panel2.Size = New
System.Drawing.Size(334, 162)
        Me.Panel2.TabIndex = 17
        '
        'Button2
        '
        Me.Button2.Location = New
System.Drawing.Point(200, 113)
        Me.Button2.Name = "Button2"
        Me.Button2.Size = New
System.Drawing.Size(75, 23)
        Me.Button2.TabIndex = 19
        Me.Button2.Text = "Cancel"
        Me.Button2.UseVisualStyleBackColor =
True
        '
        'Button1
        '
        Me.Button1.Location = New
System.Drawing.Point(52, 113)
        Me.Button1.Name = "Button1"
        Me.Button1.Size = New
System.Drawing.Size(75, 23)
        Me.Button1.TabIndex = 18
        Me.Button1.Text = "Ok"
        Me.Button1.UseVisualStyleBackColor =
True
        '
        'Panel3
        '
        Me.Panel3.BackColor =
System.Drawing.SystemColors.Control
        Me.Panel3.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle
        Me.Panel3.Controls.Add(Me.Panel1)
        Me.Panel3.Location = New
System.Drawing.Point(53, 14)
        Me.Panel3.Name = "Panel3"
        Me.Panel3.Size = New
System.Drawing.Size(222, 69)
        Me.Panel3.TabIndex = 17
        '
        'frm_numStates
        '
        Me.AutoScaleDimensions = New
System.Drawing.SizeF(6.0!, 14.0!)
        Me.AutoScaleMode =
System.Windows.Forms.AutoScaleMode.Font
        Me.AutoValidate =
System.Windows.Forms.AutoValidate.EnablePrevent
FocusChange
        Me.BackColor =
System.Drawing.SystemColors.Control
        Me.ClientSize = New
System.Drawing.Size(339, 168)
        Me.Controls.Add(Me.Panel2)
        Me.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedSingl
e
        Me.Icon =
CType(resources.GetObject("$this.Icon"),
System.Drawing.Icon)
        Me.Name = "frm_numStates"
        Me.Text = "REND"
        Me.Panel1.ResumeLayout(False)
        Me.Panel1.PerformLayout()
        Me.Panel2.ResumeLayout(False)
        Me.Panel3.ResumeLayout(False)
        Me.ResumeLayout(False)

    End Sub
    Friend WithEvents lbl_numStates As
System.Windows.Forms.Label
    Friend WithEvents cmb_numStates As
System.Windows.Forms.ComboBox
    Friend WithEvents Panel1 As
System.Windows.Forms.Panel
    Friend WithEvents Panel2 As
System.Windows.Forms.Panel
    Friend WithEvents Panel3 As
System.Windows.Forms.Panel
    Friend WithEvents Button2 As
System.Windows.Forms.Button
    Friend WithEvents Button1 As
System.Windows.Forms.Button

End Class
```

Frm_numstate.vb

```
Public Class frm_numStates
    Private P_array As New ArrayList
    Private transtable As frm_transTable
    Public P_type As String = Nothing
    Private P_parent As frm_REND = Nothing
    Public Sub New(ByVal type As String, ByVal
form As Form)

        ' This call is required by the Windows
Form Designer.

        InitializeComponent()
        P_type = type
        P_parent = form
        If P_type.Equals("DFA") Then
            Me.Text = "DFA"
            setComboBox("DFA")
        End If
        If P_type.Equals("NFA") Then
            Me.Text = "NFA"
            setComboBox("NFA")
        End If
        If P_type.Equals("NFA-E") Then
            Me.Text = "ε-NFA"
            setComboBox("NFA")
        End If
        cmb_numStates.SelectedIndex = 0
    End Sub


    Public Sub setComboBox(ByVal type As
String)
        Dim num As Integer = 0
        cmb_numStates.Items.Clear()
        If type.Equals("NFA") Then
            num = 5
        Else
            num = 10
        End If
        For i As Integer = 1 To num
            cmb_numStates.Items.Add(i)
        Next
        cmb_numStates.SelectedItem = "1"
    End Sub
    Public Sub createArrayInputs()
        P_array = New ArrayList
        If P_type.Equals("NFA-E") Then
            P_array.Add("ε")
        End If
        P_array.Add("0")
        P_array.Add("1")

    End Sub

    Private Sub btnOk_numStates_Click(ByVal
sender As System.Object, ByVal e As
System.EventArgs) Handles Button1.Click
        createArrayInputs()
        Dim numInputs = P_array.Count
        transtable = New frm_transTable(P_type,
Me, P_parent)
```

```
        transtable.setNumStates(CInt(cmb_numSta
tes.Items(cmb_numStates.SelectedIndex).ToString
))
        transtable.setNumInputs(numInputs)
        transtable.setInputs(P_array)
        transtable.drawTable()
        transtable.Style()
        transtable.Show()
        Me.Enabled = False
    End Sub


    Public ReadOnly Property
getFormTranstable() As frm_transTable
        Get
            Return transtable
        End Get
    End Property

    Private Sub btn_Cancel_Click(ByVal sender
As System.Object, ByVal e As System.EventArgs)
Handles Button2.Click
        Me.Close()
    End Sub

    Private Sub
cmb_numInputs_SelectedIndexChanged(ByVal sender
As System.Object, ByVal e As System.EventArgs)



    End Sub


    Private Sub frm_numStates_Load(ByVal sender
As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load

    End Sub


    Private Sub frm_numStates_FormClosed(ByVal
sender As System.Object, ByVal e As
System.Windows.Forms.FormClosedEventArgs)
Handles MyBase.FormClosed
        P_parent.Enabled = True
    End Sub

    Private Sub Panel4_Paint(ByVal sender As
System.Object, ByVal e As
System.Windows.Forms.PaintEventArgs)

    End Sub
End Class
```

Frm_transtable.designer.vb

```
<Global.Microsoft.VisualBasic.CompilerServices.
DesignerGenerated()> _
Partial Class frm_transTable
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the
component list.
    <System.Diagnostics.DebuggerNonUserCode()>
_
    Protected Overrides Sub Dispose(ByVal
disposing As Boolean)
        Try
            If disposing AndAlso components
IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As
System.ComponentModel.IContainer

    'NOTE: The following procedure is required
by the Windows Form Designer
    'It can be modified using the Windows Form
Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()>
_
    Private Sub InitializeComponent()
        Dim DataGridViewCellStyle1 As
System.Windows.Forms.DataGridViewCellStyle =
New System.Windows.Forms.DataGridViewCellStyle
        Dim DataGridViewCellStyle2 As
System.Windows.Forms.DataGridViewCellStyle =
New System.Windows.Forms.DataGridViewCellStyle
        Dim DataGridViewCellStyle3 As
System.Windows.Forms.DataGridViewCellStyle =
New System.Windows.Forms.DataGridViewCellStyle
        Dim DataGridViewCellStyle4 As
System.Windows.Forms.DataGridViewCellStyle =
New System.Windows.Forms.DataGridViewCellStyle
        Dim DataGridViewCellStyle5 As
System.Windows.Forms.DataGridViewCellStyle =
New System.Windows.Forms.DataGridViewCellStyle
        Dim resources As
System.ComponentModel.ComponentResourceManager
= New
System.ComponentModel.ComponentResourceManager(
GetType(frm_transTable))
        Me.DataGridViewTransTable = New
System.Windows.Forms.DataGridView
        Me.P_lblTransTable = New
System.Windows.Forms.Label
        Me.P_btnOK = New
System.Windows.Forms.Button
        Me.P_btnCancel = New
System.Windows.Forms.Button
        Me.P_pnlBack = New
System.Windows.Forms.Panel
        CType(Me.DataGridViewTransTable,
System.ComponentModel.ISupportInitialize).Begin
Init()
        Me.P_pnlBack.SuspendLayout()
        Me.SuspendLayout()
        '
        'DataGridViewTransTable
        '
        Me.DataGridViewTransTable.AllowUserToAd
dRows = False
        Me.DataGridViewTransTable.AllowUserToDe
leteRows = False
        Me.DataGridViewTransTable.AllowUserToRe
sizeColumns = False
        Me.DataGridViewTransTable.AllowUserToRe
sizeRows = False
        DataGridViewCellStyle1.BackColor =
System.Drawing.Color.DarkSeaGreen
        Me.DataGridViewTransTable.AlternatingRo
wsDefaultCellStyle = DataGridViewCellStyle1
        Me.DataGridViewTransTable.BackgroundCol
or = System.Drawing.SystemColors.Control
        Me.DataGridViewTransTable.ClipboardCopy
Mode =
System.Windows.Forms.DataGridViewClipboardCopyM
ode.Disable
        Me.DataGridViewTransTable.ColumnHeaders
BorderStyle =
System.Windows.Forms.DataGridViewHeaderBorderSt
yle.Sunken
        DataGridViewCellStyle2.Alignment =
System.Windows.Forms.DataGridViewContentAlignme
nt.MiddleCenter
        DataGridViewCellStyle2.BackColor =
System.Drawing.Color.DarkSeaGreen
        DataGridViewCellStyle2.Font = New
System.Drawing.Font("Arial Black", 9.0!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        DataGridViewCellStyle2.ForeColor =
System.Drawing.SystemColors.WindowText
        DataGridViewCellStyle2.SelectionBackCol
or = System.Drawing.SystemColors.Highlight
        DataGridViewCellStyle2.SelectionForeCol
or = System.Drawing.SystemColors.HighlightText
        DataGridViewCellStyle2.WrapMode =
System.Windows.Forms.DataGridViewTriState.
[True]
        Me.DataGridViewTransTable.ColumnHeaders
DefaultCellStyle = DataGridViewCellStyle2
        Me.DataGridViewTransTable.ColumnHeaders
HeightSizeMode =
System.Windows.Forms.DataGridViewColumnHeadersH
eightSizeMode.DisableResizing
        Me.DataGridViewTransTable.Cursor =
System.Windows.Forms.Cursors.Hand
        DataGridViewCellStyle3.Alignment =
System.Windows.Forms.DataGridViewContentAlignme
nt.MiddleCenter
        DataGridViewCellStyle3.BackColor =
System.Drawing.Color.DarkSeaGreen
        DataGridViewCellStyle3.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        DataGridViewCellStyle3.ForeColor =
System.Drawing.SystemColors.ControlText
        DataGridViewCellStyle3.SelectionBackCol
or = System.Drawing.SystemColors.Highlight
        DataGridViewCellStyle3.SelectionForeCol
or = System.Drawing.SystemColors.HighlightText
        DataGridViewCellStyle3.WrapMode =
System.Windows.Forms.DataGridViewTriState.
[False]
        Me.DataGridViewTransTable.DefaultCellSt
yle = DataGridViewCellStyle3
        Me.DataGridViewTransTable.EnableHeaders
VisualStyles = False
        Me.DataGridViewTransTable.GridColor =
System.Drawing.Color.DarkSeaGreen
        Me.DataGridViewTransTable.Location =
New System.Drawing.Point(19, 36)
        Me.DataGridViewTransTable.Margin = New
System.Windows.Forms.Padding(10)
        Me.DataGridViewTransTable.MultiSelect =
False
        Me.DataGridViewTransTable.Name =
"DataGridViewTransTable"
        DataGridViewCellStyle4.Alignment =
System.Windows.Forms.DataGridViewContentAlignme
nt.MiddleCenter
        DataGridViewCellStyle4.BackColor =
System.Drawing.Color.DarkSeaGreen
        DataGridViewCellStyle4.Font = New
System.Drawing.Font("Arial", 9.0!,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        DataGridViewCellStyle4.ForeColor =
System.Drawing.SystemColors.WindowText
        DataGridViewCellStyle4.SelectionBackCol
or = System.Drawing.SystemColors.Highlight
        DataGridViewCellStyle4.SelectionForeCol
or = System.Drawing.SystemColors.HighlightText
        DataGridViewCellStyle4.WrapMode =
System.Windows.Forms.DataGridViewTriState.
[True]
        Me.DataGridViewTransTable.RowHeadersDef
aultCellStyle = DataGridViewCellStyle4
        Me.DataGridViewTransTable.RowHeadersWid
thSizeMode =
System.Windows.Forms.DataGridViewRowHeadersWidt
hSizeMode.DisableResizing
        DataGridViewCellStyle5.BackColor =
System.Drawing.Color.DarkSeaGreen
        Me.DataGridViewTransTable.RowsDefaultCe
llStyle = DataGridViewCellStyle5
        Me.DataGridViewTransTable.RowTemplate.D
efaultCellStyle.BackColor =
System.Drawing.Color.White
        Me.DataGridViewTransTable.ScrollBars =
System.Windows.Forms.ScrollBars.None
        Me.DataGridViewTransTable.SelectionMode
=
System.Windows.Forms.DataGridViewSelectionMode.
CellSelect
        Me.DataGridViewTransTable.ShowEditingIc
on = False
        Me.DataGridViewTransTable.Size = New
System.Drawing.Size(207, 153)
        Me.DataGridViewTransTable.TabIndex = 0
        '
        'P_lblTransTable
        '
        Me.P_lblTransTable.AutoSize = True
        Me.P_lblTransTable.BackColor =
System.Drawing.Color.DarkSeaGreen
        Me.P_lblTransTable.Font = New
System.Drawing.Font("Arial Black", 8.25!,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.P_lblTransTable.Location = New
System.Drawing.Point(12, 9)
        Me.P_lblTransTable.Name =
"P_lblTransTable"
        Me.P_lblTransTable.Size = New
System.Drawing.Size(129, 15)
        Me.P_lblTransTable.TabIndex = 1
        Me.P_lblTransTable.Text = "TRANSITION
TABLE"
        '
        'P_btnOK
        '
        Me.P_btnOK.BackColor =
System.Drawing.Color.DarkSeaGreen
        Me.P_btnOK.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.P_btnOK.Location = New
System.Drawing.Point(16, 197)
        Me.P_btnOK.Name = "P_btnOK"
        Me.P_btnOK.Size = New
System.Drawing.Size(75, 23)
        Me.P_btnOK.TabIndex = 2
        Me.P_btnOK.Text = "Ok"
        Me.P_btnOK.UseVisualStyleBackColor =
True
        '
        'P_btnCancel
        '
        Me.P_btnCancel.BackColor =
System.Drawing.Color.DarkSeaGreen
        Me.P_btnCancel.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.P_btnCancel.Location = New
System.Drawing.Point(148, 197)
        Me.P_btnCancel.Name = "P_btnCancel"
        Me.P_btnCancel.Size = New
System.Drawing.Size(75, 23)
        Me.P_btnCancel.TabIndex = 3
        Me.P_btnCancel.Text = "Cancel"
        Me.P_btnCancel.UseVisualStyleBackColor
= True
        '
        'P_pnlBack
        '
        Me.P_pnlBack.BackColor =
System.Drawing.Color.DarkSeaGreen
        Me.P_pnlBack.Controls.Add(Me.P_btnCance
l)
        Me.P_pnlBack.Controls.Add(Me.P_btnOK)
        Me.P_pnlBack.Location = New
System.Drawing.Point(3, 3)
        Me.P_pnlBack.Name = "P_pnlBack"
        Me.P_pnlBack.Size = New
System.Drawing.Size(230, 229)
        Me.P_pnlBack.TabIndex = 4
        '
        'frm_transTable
        '
        Me.AutoScaleDimensions = New
System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode =
System.Windows.Forms.AutoScaleMode.Font
        Me.AutoValidate =
System.Windows.Forms.AutoValidate.EnableAllowFo
cusChange
        Me.BackColor =
System.Drawing.SystemColors.Control
        Me.ClientSize = New
System.Drawing.Size(237, 235)
        Me.Controls.Add(Me.P_lblTransTable)
        Me.Controls.Add(Me.DataGridViewTransTab
le)
        Me.Controls.Add(Me.P_pnlBack)
        Me.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedSingl
e
        Me.Icon =
CType(resources.GetObject("$this.Icon"),
System.Drawing.Icon)
        Me.Name = "frm_transTable"
        Me.StartPosition =
System.Windows.Forms.FormStartPosition.CenterPa
rent
        Me.Text = "frm_transTable"
        CType(Me.DataGridViewTransTable,
System.ComponentModel.ISupportInitialize).EndIn
it()
        Me.P_pnlBack.ResumeLayout(False)
        Me.ResumeLayout(False)
        Me.PerformLayout()

    End Sub
    Friend WithEvents DataGridViewTransTable As
System.Windows.Forms.DataGridView
    Friend WithEvents P_lblTransTable As
System.Windows.Forms.Label
    Friend WithEvents P_btnOK As
System.Windows.Forms.Button
    Friend WithEvents P_btnCancel As
System.Windows.Forms.Button
    Friend WithEvents P_pnlBack As
System.Windows.Forms.Panel
End Class
```

Frm_transtable.vb

```
Public Class frm_transTable

    Private P_form As Form
    Private P_parent As frm_REND

    Private P_numStates As Integer
    Private P_numInputs As Integer
    Private P_array As New ArrayList
    Private P_tableComp As New ArrayList
    Private P_States As New ArrayList
    Private P_type As String

    Private column0 As dataGridColumns
    Private column1 As dataGridColumns
    Private column2 As dataGridColumns
    Private column3 As dataGridColumns
    Private column4 As dataGridColumns
    Private column5 As dataGridColumns
    Private column6 As dataGridColumns

    Public Sub New(ByVal type As String, ByVal
f As Form, ByVal _parent As frm_REND)
        Me.SetStyle(ControlStyles.UserPaint Or
ControlStyles.AllPaintingInWmPaint, True)
        ' This call is required by the Windows
Form Designer.
        InitializeComponent()
        ' Add any initialization after the
InitializeComponent() call.
        P_type = type
        If P_type.Equals("DFA") Then
            Me.Text = "DFA"
        End If
        If P_type.Equals("NFA") Then
            Me.Text = "NFA"
        End If
        If P_type.Equals("NFA-E") Then
            Me.Text = "ε-NFA"
        End If

        P_form = f
        P_parent = _parent

    End Sub

    Public Sub setNumStates(ByVal num As
Integer)
        P_numStates = num
    End Sub

    Public Sub setNumInputs(ByVal num As
Integer)
        P_numInputs = num
    End Sub

    Public Sub setInputs(ByVal arrInputs As
ArrayList)
        P_array = arrInputs
    End Sub

    Public Sub Style()
        DataGridViewTransTable.Columns(0).Width
= 60
        DataGridViewTransTable.RowHeadersWidth
= 60
        If P_type.Equals("DFA") Then
            For i As Integer = 1 To
DataGridViewTransTable.Columns.Count - 1
                DataGridViewTransTable.Columns(
i).Width = 50
            Next
            DataGridViewTransTable.Width =
DataGridViewTransTable.Columns(0).Width + _
                                            Data
GridViewTransTable.Columns(1).Width *
DataGridViewTransTable.Columns.Count - 1 _
                                             +
13
            DataGridViewTransTable.Height =
DataGridViewTransTable.Rows(0).Height + _
                                            Data
GridViewTransTable.Rows.Item(0).Height *
DataGridViewTransTable.Rows.Count - 1 _
                                             + 3
            Me.Width =
DataGridViewTransTable.Width + 50
            Me.Height =
DataGridViewTransTable.Height + 130
            Me.P_btnOK.Location = New
Point(Me.Width / 2 - P_btnOK.Width - 10,
DataGridViewTransTable.Height + 50)
            Me.P_btnCancel.Location = New
Point(Me.Width / 2 + 10,
DataGridViewTransTable.Height + 50)
            Me.P_pnlBack.Width = Me.Width - 12
            Me.P_pnlBack.Height = Me.Height -
32
        End If

        Dim width As Double = 0
        Dim span As Double = 0

        Select Case P_numStates
            Case Is = 1
                width = 50
                span = 13
            Case Is = 2
                width = 65
                span = -3
            Case Is = 3
                width = 80
                span = -16
            Case Is = 4
                width = 95
                span = -31
            Case Is = 5
                width = 110
                span = -47
            Case Is = 6
                width = 125
                span = -62
            Case Is = 7
                width = 140
                span = -78
            Case Is = 8
                width = 155
                span = -93
            Case Is = 9
                width = 170
                span = -109
            Case Is = 10
                width = 185
                span = -124

        End Select

        If P_type.Equals("NFA-E") Or
P_type.Equals("NFA") Then
            For i As Integer = 1 To
DataGridViewTransTable.Columns.Count - 1
                DataGridViewTransTable.Columns(
i).Width = width
            Next
            DataGridViewTransTable.Width =
DataGridViewTransTable.Columns(0).Width + _
                                               Data
GridViewTransTable.Columns.Item(1).Width *
DataGridViewTransTable.Columns.Count - 1 _
                                                +
span
            DataGridViewTransTable.Height =
DataGridViewTransTable.Rows(0).Height + _
                                               Data
GridViewTransTable.Rows.Item(0).Height *
DataGridViewTransTable.Rows.Count - 1 _
                                                + 3
            Me.Width =
DataGridViewTransTable.Width + 50
            Me.Height =
DataGridViewTransTable.Height + 130
            Me.P_btnOK.Location = New
Point(Me.Width / 2 - P_btnOK.Width - 10,
DataGridViewTransTable.Height + 50)
            Me.P_btnCancel.Location = New
Point(Me.Width / 2 + 10,
DataGridViewTransTable.Height + 50)
            Me.P_pnlBack.Width = Me.Width - 12
            Me.P_pnlBack.Height = Me.Height -
32
        End If
    End Sub

    Public Sub drawTable()
        column0 = New dataGridColumns(P_type)
        column1 = New dataGridColumns(P_type)
        column2 = New dataGridColumns(P_type)
        column3 = New dataGridColumns(P_type)
        column4 = New dataGridColumns(P_type)
        column5 = New dataGridColumns(P_type)
        column6 = New dataGridColumns(P_type)
        With DataGridViewTransTable.Columns
            With column0
                .setStartFinal()
                .setHeader("State")
            End With
            .Add(column0.getDataGridColumn())
            If P_numInputs >= 1 Then
                With column1
                    .setNumState(P_numStates)
                    .setMembers()
                    .setHeader(P_array.Item(0))
                End With
                .Add(column1.getDataGridColumn(
))
            End If

            If P_numInputs >= 2 Then
                With column2
                    .setNumState(P_numStates)
                    .setMembers()
                    .setHeader(P_array.Item(1))
                End With
                .Add(column2.getDataGridColumn(
))
            End If

            If P_numInputs >= 3 Then
                With column3
                    .setNumState(P_numStates)
                    .setMembers()
                    .setHeader(P_array.Item(2))
                End With
                .Add(column3.getDataGridColumn(
))
            End If

            If P_numInputs >= 4 Then
                With column4
                    .setNumState(P_numStates)
                    .setMembers()
                    .setHeader(P_array.Item(3))
                End With
                .Add(column4.getDataGridColumn(
))
            End If

            If P_numInputs >= 5 Then
                With column5
                    .setNumState(P_numStates)
                    .setMembers()
                    .setHeader(P_array.Item(4))
                End With
                .Add(column5.getDataGridColumn(
))
            End If
            If P_numInputs >= 6 Then
                With column6
                    .setNumState(P_numStates)
                    .setMembers()
                    .setHeader(P_array.Item(5))
                End With
                .Add(column6.getDataGridColumn(
))
            End If

        End With

        DataGridViewTransTable.Rows.Add(P_numSt
ates)
        With DataGridViewTransTable.Rows
            For i As Integer = 0 To P_numStates
- 1
                .Item(i).HeaderCell.Value = "q"
& i.ToString
            Next
        End With
        initializeTable()

    End Sub

    Public Sub initializeTable()
        For col0 As Integer = 0 To 0
```

```
            For row0 As Integer = 0 To
P_numStates - 1
                DataGridViewTransTable.Item(col
0, row0).Value = "Normal"
            Next
        Next

        For col As Integer = 1 To P_numInputs
            For row As Integer = 0 To
P_numStates - 1
                DataGridViewTransTable.Item(col
, row).Value = "ø"
            Next
        Next


    End Sub


    Private Sub frm_transTable_Load(ByVal
sender As System.Object, ByVal e As
System.EventArgs) Handles MyBase.Load
    End Sub



    Private Sub P_btnOK_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles P_btnOK.Click
        If checkStartState() And
checkNumStartState() And checkFinalState() And
checkEcLose() Then
            If P_type.Equals("DFA") Then
                Dim m_DFA As New
DFA(DataGridViewTransTable, "create")
                P_parent.setTable(m_DFA.Table)
                P_parent.setEvents(m_DFA.getCom
mands)
                P_parent.setNumstate(P_numState
s)
                P_parent.setArrInputs(m_DFA.get
Inputs)
                P_parent.setMiniTable(m_DFA.Tab
le)
            End If
            If P_type.Equals("NFA") Or
P_type.Equals("NFA-E") Then
                Dim m_NFA As New
NFA(DataGridViewTransTable)
                P_parent.setEvents(m_NFA.getCom
mands)
                P_parent.setNumstate(P_numState
s)
                P_parent.setArrInputs(m_NFA.get
Inputs)
            End If
            P_parent.setTable(DataGridViewTrans
Table)
            P_parent.setOriginalTable(DataGridV
iewTransTable)
            P_parent.setMiniTable(DataGridViewT
ransTable)
            Me.Close()
            P_form.Close()
        Else
            Dim Err = "     1. No Start State
specified." & _
                   vbCrLf & "     2. Too many
start States." & _
                   vbCrLf & "     3. No Final
State specified."
            If P_type.Equals("NFA-E") Then
                Err = Err & vbCrLf & "     4.
Eclose of a State does not include itself."
            End If
            Dim m As New Message("Error!",
"Processing Failed", "Cannot process wrong
input.", "These are the possible error/s:" &
vbCrLf & Err, Me)
            m.Show()
        End If


    End Sub

    Public Function checkEclose() As Boolean
        If P_type.Equals("NFA") Or
P_type.Equals("DFA") Then
            Return True
        Else
            For i As Integer = 0 To
DataGridViewTransTable.Rows.Count - 1
                Dim x As String =
DataGridViewTransTable.Item(1, i).Value
                If Not
x.Contains(DataGridViewTransTable.Rows(i).Heade
rCell.Value) Then
                    Return False
                    Exit For
                End If
            Next
        End If
        Return True
    End Function

    Public Function checkStartState() As
Boolean
        For i As Integer = 0 To
DataGridViewTransTable.Rows.Count - 1
            If DataGridViewTransTable.Item(0,
i).Value = "Start" Or
DataGridViewTransTable.Item(0, i).Value =
"Start&Final" Then
                Return True
                Exit For
            End If
        Next
        Return False
    End Function

    Public Function checkNumStartState() As
Boolean
        Dim num As Integer = 0
        For i As Integer = 0 To
DataGridViewTransTable.Rows.Count - 1
            If DataGridViewTransTable.Item(0,
i).Value = "Start" Or
DataGridViewTransTable.Item(0, i).Value =
"Start&Final" Then
                num = num + 1
            End If
        Next
        If num = 1 Then
            Return True
        Else
            Return False
        End If

    End Function
    Public Function checkFinalState() As
Boolean
        For i As Integer = 0 To
DataGridViewTransTable.Rows.Count - 1
            If DataGridViewTransTable.Item(0,
i).Value = "Final" Or
DataGridViewTransTable.Item(0, i).Value =
"Start&Final" Then
                Return True
                Exit For
            End If
        Next
        Return False
    End Function

    Public ReadOnly Property statesArray() As
ArrayList
        Get
            Return P_States
        End Get
    End Property
    Private Sub frm_transTable_FormClosed(ByVal
sender As System.Object, ByVal e As
System.Windows.Forms.FormClosedEventArgs)
Handles MyBase.FormClosed
        P_form.Enabled = True
    End Sub

    Private Sub P_lblTransTable_Click(ByVal
sender As System.Object, ByVal e As
System.EventArgs) Handles P_lblTransTable.Click

    End Sub

    Private Sub P_btnCancel_Click(ByVal sender
As System.Object, ByVal e As System.EventArgs)
Handles P_btnCancel.Click
        P_form.Enabled = True
        Me.Close()
    End Sub
End Class
```

Frm_RE.designer.vb

```
<Global.Microsoft.VisualBasic.CompilerServices.
DesignerGenerated()> _
Partial Class frm_RE
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the
component list.
    <System.Diagnostics.DebuggerNonUserCode()>
_
    Protected Overrides Sub Dispose(ByVal
disposing As Boolean)
        Try
            If disposing AndAlso components
IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As
System.ComponentModel.IContainer

    'NOTE: The following procedure is required
by the Windows Form Designer
    'It can be modified using the Windows Form
Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()>
_
    Private Sub InitializeComponent()
        Dim resources As
System.ComponentModel.ComponentResourceManager
= New
System.ComponentModel.ComponentResourceManager(
GetType(frm_RE))
        Me.Label1 = New
System.Windows.Forms.Label
        Me.TextBox1 = New
System.Windows.Forms.TextBox
        Me.Button1 = New
System.Windows.Forms.Button
        Me.Button2 = New
System.Windows.Forms.Button
        Me.Label2 = New
System.Windows.Forms.Label
        Me.Label3 = New
System.Windows.Forms.Label
        Me.Label4 = New
System.Windows.Forms.Label
        Me.ComboBox1 = New
System.Windows.Forms.ComboBox
        Me.ComboBox2 = New
System.Windows.Forms.ComboBox
        Me.ComboBox3 = New
System.Windows.Forms.ComboBox
        Me.Panel1 = New
System.Windows.Forms.Panel
        Me.Label5 = New
System.Windows.Forms.Label
        Me.ComboBox4 = New
System.Windows.Forms.ComboBox
        Me.Panel2 = New
System.Windows.Forms.Panel
        Me.Panel1.SuspendLayout()
        Me.SuspendLayout()
        '
        'Label1
        '
        Me.Label1.AutoSize = True
        Me.Label1.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.Label1.Location = New
System.Drawing.Point(70, 161)
        Me.Label1.Name = "Label1"
        Me.Label1.Size = New
System.Drawing.Size(146, 14)
        Me.Label1.TabIndex = 0
        Me.Label1.Text = "Regular Expression
{0,1} :"
        '
        'TextBox1
        '
        Me.TextBox1.ForeColor =
System.Drawing.Color.DarkRed
        Me.TextBox1.Location = New
System.Drawing.Point(56, 178)
        Me.TextBox1.Name = "TextBox1"
        Me.TextBox1.ReadOnly = True
        Me.TextBox1.Size = New
System.Drawing.Size(172, 20)
        Me.TextBox1.TabIndex = 1
        '
        'Button1
        '
        Me.Button1.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.Button1.Location = New
System.Drawing.Point(56, 204)
        Me.Button1.Name = "Button1"
        Me.Button1.Size = New
System.Drawing.Size(75, 23)
        Me.Button1.TabIndex = 2
        Me.Button1.Text = "Ok"
        Me.Button1.UseVisualStyleBackColor =
True
        '
        'Button2
        '
        Me.Button2.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.Button2.Location = New
System.Drawing.Point(153, 204)
        Me.Button2.Name = "Button2"
        Me.Button2.Size = New
System.Drawing.Size(75, 23)
        Me.Button2.TabIndex = 3
        Me.Button2.Text = "Cancel"
        Me.Button2.UseVisualStyleBackColor =
True
        '
        'Label2
        '
        Me.Label2.AutoSize = True
        Me.Label2.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.Label2.Location = New
System.Drawing.Point(25, 13)
        Me.Label2.Name = "Label2"
        Me.Label2.Size = New
System.Drawing.Size(49, 14)
        Me.Label2.TabIndex = 7
        Me.Label2.Text = "1st Term"
        '
        'Label3
        '
        Me.Label3.AutoSize = True
        Me.Label3.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.Label3.Location = New
System.Drawing.Point(25, 41)
        Me.Label3.Name = "Label3"
        Me.Label3.Size = New
System.Drawing.Size(52, 14)
        Me.Label3.TabIndex = 8
        Me.Label3.Text = "2nd Term"
        '
        'Label4
        '
        Me.Label4.AutoSize = True
        Me.Label4.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.Label4.Location = New
System.Drawing.Point(26, 69)
        Me.Label4.Name = "Label4"
        Me.Label4.Size = New
System.Drawing.Size(50, 14)
        Me.Label4.TabIndex = 9
        Me.Label4.Text = "3rd Term"
        '
        'ComboBox1
        '
        Me.ComboBox1.DropDownStyle =
System.Windows.Forms.ComboBoxStyle.DropDownList
        Me.ComboBox1.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.ComboBox1.FormattingEnabled = True
        Me.ComboBox1.Items.AddRange(New
Object() {"0", "1", "0*", "1*", "(0+1)",
"(0+1)*"})
        Me.ComboBox1.Location = New
System.Drawing.Point(137, 33)
        Me.ComboBox1.Name = "ComboBox1"
        Me.ComboBox1.Size = New
System.Drawing.Size(78, 22)
        Me.ComboBox1.TabIndex = 10
        '
        'ComboBox2
        '
        Me.ComboBox2.DropDownStyle =
System.Windows.Forms.ComboBoxStyle.DropDownList
        Me.ComboBox2.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.ComboBox2.FormattingEnabled = True
        Me.ComboBox2.Items.AddRange(New
Object() {"0", "1", "0*", "1*", "(0+1)",
"(0+1)*"})
        Me.ComboBox2.Location = New
System.Drawing.Point(137, 61)
        Me.ComboBox2.Name = "ComboBox2"
        Me.ComboBox2.Size = New
System.Drawing.Size(78, 22)
        Me.ComboBox2.TabIndex = 11
        '
        'ComboBox3
        '
        Me.ComboBox3.DropDownStyle =
System.Windows.Forms.ComboBoxStyle.DropDownList
        Me.ComboBox3.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.ComboBox3.FormattingEnabled = True
        Me.ComboBox3.Items.AddRange(New
Object() {"0", "1", "0*", "1*", "(0+1)",
"(0+1)*"})
        Me.ComboBox3.Location = New
System.Drawing.Point(136, 89)
        Me.ComboBox3.Name = "ComboBox3"
        Me.ComboBox3.Size = New
System.Drawing.Size(79, 22)
        Me.ComboBox3.TabIndex = 12
        '
        'Panel1
        '
        Me.Panel1.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle
        Me.Panel1.Controls.Add(Me.Label5)
        Me.Panel1.Controls.Add(Me.ComboBox4)
        Me.Panel1.Controls.Add(Me.Label2)
        Me.Panel1.Controls.Add(Me.Label3)
        Me.Panel1.Controls.Add(Me.Label4)
        Me.Panel1.Location = New
System.Drawing.Point(43, 22)
        Me.Panel1.Name = "Panel1"
        Me.Panel1.Size = New
System.Drawing.Size(200, 125)
        Me.Panel1.TabIndex = 13
        '
        'Label5
        '
        Me.Label5.AutoSize = True
        Me.Label5.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.Label5.Location = New
System.Drawing.Point(26, 94)
        Me.Label5.Name = "Label5"
        Me.Label5.Size = New
System.Drawing.Size(49, 14)
        Me.Label5.TabIndex = 15
        Me.Label5.Text = "4th Term"
        '
        'ComboBox4
        '
        Me.ComboBox4.DropDownStyle =
System.Windows.Forms.ComboBoxStyle.DropDownList
        Me.ComboBox4.FormattingEnabled = True
        Me.ComboBox4.Items.AddRange(New
Object() {"0", "1", "0*", "1*", "(0+1)",
"(0+1)*"})
        Me.ComboBox4.Location = New
System.Drawing.Point(93, 94)
        Me.ComboBox4.Name = "ComboBox4"
        Me.ComboBox4.Size = New
System.Drawing.Size(78, 21)
        Me.ComboBox4.TabIndex = 10
        '
        'Panel2
        '
        Me.Panel2.BackColor =
System.Drawing.SystemColors.ButtonFace
        Me.Panel2.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle
        Me.Panel2.Location = New
System.Drawing.Point(40, 20)
        Me.Panel2.Name = "Panel2"
        Me.Panel2.Size = New
System.Drawing.Size(206, 130)
        Me.Panel2.TabIndex = 14
        '
        'frm_RE
        '
        Me.AutoScaleDimensions = New
System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode =
System.Windows.Forms.AutoScaleMode.Font
        Me.BackColor =
System.Drawing.Color.DarkSeaGreen
        Me.ClientSize = New
System.Drawing.Size(284, 233)
        Me.Controls.Add(Me.ComboBox3)
        Me.Controls.Add(Me.ComboBox2)
        Me.Controls.Add(Me.ComboBox1)
        Me.Controls.Add(Me.Button2)
        Me.Controls.Add(Me.Button1)
        Me.Controls.Add(Me.TextBox1)
        Me.Controls.Add(Me.Label1)
        Me.Controls.Add(Me.Panel1)
        Me.Controls.Add(Me.Panel2)
        Me.Icon =
CType(resources.GetObject("$this.Icon"),
System.Drawing.Icon)
        Me.Name = "frm_RE"
        Me.Text = "Regular Expression"
        Me.Panel1.ResumeLayout(False)
        Me.Panel1.PerformLayout()
        Me.ResumeLayout(False)
        Me.PerformLayout()

    End Sub
    Friend WithEvents Label1 As
System.Windows.Forms.Label
    Friend WithEvents TextBox1 As
System.Windows.Forms.TextBox
    Friend WithEvents Button1 As
System.Windows.Forms.Button
    Friend WithEvents Button2 As
System.Windows.Forms.Button
    Friend WithEvents Label2 As
System.Windows.Forms.Label
    Friend WithEvents Label3 As
System.Windows.Forms.Label
    Friend WithEvents Label4 As
System.Windows.Forms.Label
    Friend WithEvents ComboBox1 As
System.Windows.Forms.ComboBox
    Friend WithEvents ComboBox2 As
System.Windows.Forms.ComboBox
    Friend WithEvents ComboBox3 As
System.Windows.Forms.ComboBox
    Friend WithEvents Panel1 As
System.Windows.Forms.Panel
    Friend WithEvents Panel2 As
System.Windows.Forms.Panel
    Friend WithEvents Label5 As
System.Windows.Forms.Label
    Friend WithEvents ComboBox4 As
System.Windows.Forms.ComboBox
End Class
```

Frm_RE.vb

```
Public Class frm_RE
    Private Commands As New ArrayList
    Private arrayofInputs As New ArrayList
    Private arrayOfArrows As New ArrayList
    Private posX As Double = -22
    Private posY As Double = 6
    Private dir As String = "right"
    Private statetype As String = ""
    Private index As Integer = 0
    Private P_parent As frm_REND
    Private numstate As Integer = 0
    Private down As Boolean = False


    Public Sub New(ByVal f As frm_REND)
        ' This call is required by the Windows
Form Designer.
        InitializeComponent()
        Commands = New ArrayList
        arrayofInputs = New ArrayList
        arrayOfArrows = New ArrayList
        P_parent = f
        posX = -22
        posY = 6
        ' Add any initialization after the
InitializeComponent() call.

    End Sub
    Private Sub frm_RE_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        ComboBox1.SelectedItem = "0"
        ComboBox2.SelectedItem = "0"
        ComboBox3.SelectedItem = "0"
        ComboBox4.SelectedItem = "0"
    End Sub


    Private Sub TextBox1_TextChanged(ByVal
sender As System.Object, ByVal e As
System.EventArgs) Handles TextBox1.TextChanged
        Dim z As Array =
TextBox1.Text.ToCharArray
        For i As Integer = 0 To
TextBox1.Text.Length - 1
            If Not z(i).ToString.Equals("1")
And Not z(i).ToString.Equals("0") And Not
z(i).ToString.Equals("(") _
                    And Not
z(i).ToString.Equals(")") And Not
z(i).ToString.Equals("*") And Not
z(i).ToString.Equals("+") Then
                TextBox1.Text =
TextBox1.Text.Replace(z(i).ToString, "")
                TextBox1.SelectionStart =
TextBox1.Text.Length
                TextBox1.SelectionLength = 0
                TextBox1.ScrollToCaret()
            End If
        Next
    End Sub

    Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
        processEvents()
        P_parent.setEvents(Commands)
        P_parent.setNumstate(numstate)
        P_parent.setArrInputs(arrayofInputs)
        P_parent.Enabled = True
        Me.Close()
        P_parent.setText(vbCrLf & "Auto-
conversion to ε-NFA...Success!")
        Dim re As New RegEx
        re.firstTerm =
ComboBox1.SelectedItem.ToString
        re.secondTerm =
ComboBox2.SelectedItem.ToString
        re.thirdTerm =
ComboBox3.SelectedItem.ToString
        re.fourthTerm =
ComboBox4.SelectedItem.ToString
        P_parent.setRE(re)
        P_parent.setTextRE("Clear")
        P_parent.setTextRE("The Regular
Expression: " & re.firstTerm & re.secondTerm &
re.thirdTerm & re.fourthTerm & " has been
converted to NFA." & _
                           vbCrLf & "Click view
diagram to see.")
        Dim m As Message = New
Message("Success", "Auto-Conversion
Successful!", "Conversion from Regular
Expression to ε-NFA", "Click view diagram to
see transition diagram", P_parent)
        m.Show()
    End Sub
    Public Sub processEvents()
        statetype = "start"
        processStatesCombo(ComboBox1)
        statetype = "mid"
        processStatesCombo(ComboBox2)
        statetype = "mid"
        If
ComboBox1.SelectedItem.ToString.Length > 2 And
ComboBox2.SelectedItem.ToString.Length > 2 _
           Or
ComboBox1.SelectedItem.ToString.Length > 2 And
ComboBox2.SelectedItem.ToString.Length <= 2 And
ComboBox3.SelectedItem.ToString.Length >= 2 _
           Or
ComboBox1.SelectedItem.ToString.Length <= 2 And
ComboBox2.SelectedItem.ToString.Length > 2 And
ComboBox3.SelectedItem.ToString.Length >= 2 _
           Or
ComboBox1.SelectedItem.ToString.Length = 2 And
ComboBox2.SelectedItem.ToString.Length <= 2 And
ComboBox3.SelectedItem.ToString.Length > 2 _
           Or
ComboBox1.SelectedItem.ToString.Length <= 2 And
ComboBox2.SelectedItem.ToString.Length = 2 And
ComboBox3.SelectedItem.ToString.Length > 2 Then
            posY = posY - 12
            posX = 22
            dir = "left"
            down = True
        End If
        processStatesCombo(ComboBox3)
        statetype = "final"
        processStatesCombo(ComboBox4)
        addArrowEvents()
    End Sub

    Public Sub addArrowEvents()
        For i As Integer = 0 To
arrayOfArrows.Count - 1
            Commands.Add(arrayOfArrows(i))
        Next
    End Sub

    Public Sub processStatesCombo(ByVal combo
As ComboBox)
        Dim item As String =
combo.SelectedItem.ToString
        Select Case item
            Case "0"
                NFASingle("0", dir)
            Case "1"
                NFASingle("1", dir)
            Case "0*"
                NFASingleStar("0", dir)
            Case "1*"
                NFASingleStar("1", dir)
            Case "(0+1)"
                NFAUnion(dir)
            Case "(0+1)*"
                NFAUnionStar(dir)
        End Select
    End Sub

    Private Sub NFASingle(ByVal text As String,
ByVal direction As String)
        If Not statetype.Equals("start") Then
            Dim mevent = New myEvent
            If direction.Equals("left") Then
                mevent.m_ArrowDir = "left"
                mevent.m_Arrow = "q" & index &
"to" & "q" & (index - 1)
            Else
                mevent.m_Arrow = "q" & (index -
1) & "to" & "q" & index
            End If

            If down Then
                mevent.Command =
"DrawObscureArrow"
                mevent.m_Arrow = "q" & (index -
1) & "to" & "q" & index
                down = False
            Else
                mevent.Command = "DrawLine"
            End If
            mevent.MustDo = True
            arrayOfArrows.Add(mevent)
            arrayofInputs.Add("ε")
        End If
        Dim m_event As myEvent
        Dim state1 As New State
        state1.x_Start1 = posX
        state1.y_Start1 = posY
        state1.radius = 0.75
        state1.Ref = "q" & index
        If statetype.Equals("start") Then
            state1.color = Color.Firebrick
        End If
        m_event = New myEvent
        m_event.Command = "DrawState"
        m_event.m_State = state1
        m_event.MustDo = True
        Commands.Add(m_event)
        If direction.Equals("left") Then
            posX = posX - 4
        Else
            posX = posX + 4
        End If

        index = index + 1
        numstate = numstate + 1

        m_event = New myEvent
        m_event.Command = "DrawLine"
        If direction.Equals("left") Then
            m_event.m_ArrowDir = "left"
            m_event.m_Arrow = "q" & index &
"to" & "q" & (index - 1)
        Else
            m_event.m_Arrow = "q" & (index - 1)
& "to" & "q" & index
        End If

        m_event.MustDo = True
        arrayOfArrows.Add(m_event)
        arrayofInputs.Add(text)

        Dim state2 As New State
        state2.x_Start1 = posX
        state2.y_Start1 = posY
        state2.radius = 0.75
        state2.Ref = "q" & index
        If statetype.Equals("final") Then
            state2.color = Color.Blue
        End If
        m_event = New myEvent
        m_event.Command = "DrawState"
        m_event.m_State = state2
        m_event.MustDo = True
        Commands.Add(m_event)
        If direction.Equals("left") Then
            posX = posX - 4
        Else
            posX = posX + 4
        End If
        index = index + 1
        numstate = numstate + 1

    End Sub

    Private Sub NFASingleStar(ByVal text As
String, ByVal direction As String)
        If Not statetype.Equals("start") Then
            Dim mevent = New myEvent
            If direction.Equals("left") Then
                mevent.m_ArrowDir = "left"
                mevent.m_Arrow = "q" & index &
"to" & "q" & (index - 1)
            Else
                mevent.m_Arrow = "q" & (index -
1) & "to" & "q" & index
            End If
            If down Then
                mevent.Command =
"DrawObscureArrow"
                mevent.m_Arrow = "q" & (index -
1) & "to" & "q" & index
                down = False
            Else
                mevent.Command = "DrawLine"
            End If


            mevent.MustDo = True
            arrayOfArrows.Add(mevent)
            arrayofInputs.Add("ε")
        End If
        Dim m_event As myEvent
        Dim state1 As New State
        state1.x_Start1 = posX
        state1.y_Start1 = posY
        state1.radius = 0.75
        state1.Ref = "q" & index
        If statetype.Equals("start") Then
            state1.color = Color.Firebrick
        End If
        m_event = New myEvent
        m_event.Command = "DrawState"
        m_event.m_State = state1
        m_event.MustDo = True
        Commands.Add(m_event)
        If direction.Equals("left") Then
            posX = posX - 4
        Else
            posX = posX + 4
        End If
        index = index + 1
        numstate = numstate + 1

        m_event = New myEvent
        m_event.Command = "DrawLine"
        If direction.Equals("left") Then
            m_event.m_ArrowDir = "left"
            m_event.m_Arrow = "q" & index &
"to" & "q" & (index - 1)
        Else
            m_event.m_Arrow = "q" & (index - 1)
& "to" & "q" & index
        End If

        m_event.MustDo = True
        arrayOfArrows.Add(m_event)
        arrayofInputs.Add("ε")

        m_event = New myEvent
        m_event.m_obs = 50
        m_event.Command = "DrawObscureArrow"
        If direction.Equals("left") Then
            m_event.m_ArrowDir = "left"
            m_event.m_Arrow = "q" & (index - 1)
& "to" & "q" & (index + 2)
        Else
            m_event.m_Arrow = "q" & (index - 1)
& "to" & "q" & (index + 2)
        End If

        m_event.MustDo = True
        arrayOfArrows.Add(m_event)
        arrayofInputs.Add("ε")

        Dim state2 As New State
        state2.x_Start1 = posX
        state2.y_Start1 = posY
        state2.radius = 0.75
        state2.Ref = "q" & index
        m_event = New myEvent
        m_event.Command = "DrawState"
        m_event.m_State = state2
        m_event.MustDo = True
        Commands.Add(m_event)
        If direction.Equals("left") Then
            posX = posX - 4
        Else
            posX = posX + 4
        End If
        index = index + 1
        numstate = numstate + 1

        m_event = New myEvent
        m_event.Command = "DrawLine"
        If direction.Equals("left") Then
            m_event.m_ArrowDir = "left"
            m_event.m_Arrow = "q" & index &
"to" & "q" & (index - 1)
        Else
            m_event.m_Arrow = "q" & (index - 1)
& "to" & "q" & index
        End If

        m_event.MustDo = True
        arrayOfArrows.Add(m_event)
        arrayofInputs.Add(text)

        m_event = New myEvent
        m_event.m_obs = 40
        m_event.Command = "DrawObscureArrow"
        If direction.Equals("left") Then
            m_event.m_ArrowDir = "left"
            m_event.m_Arrow = "q" & index &
"to" & "q" & (index - 1)
        Else
            m_event.m_Arrow = "q" & index &
"to" & "q" & (index - 1)
        End If

        m_event.MustDo = True
        arrayOfArrows.Add(m_event)
        arrayofInputs.Add("ε")



        Dim state3 As New State
        state3.x_Start1 = posX
        state3.y_Start1 = posY
        state3.radius = 0.75
        state3.Ref = "q" & index
        m_event = New myEvent
        m_event.Command = "DrawState"
        m_event.m_State = state3
        m_event.MustDo = True
        Commands.Add(m_event)
        If direction.Equals("left") Then
            posX = posX - 4
        Else
            posX = posX + 4
        End If
        index = index + 1
        numstate = numstate + 1


        m_event = New myEvent
        m_event.Command = "DrawLine"
        If direction.Equals("left") Then
            m_event.m_ArrowDir = "left"
            m_event.m_Arrow = "q" & index &
"to" & "q" & (index - 1)
        Else
            m_event.m_Arrow = "q" & (index - 1)
& "to" & "q" & index
        End If

        m_event.MustDo = True
        arrayOfArrows.Add(m_event)
        arrayofInputs.Add("ε")


        Dim state4 As New State
        state4.x_Start1 = posX
        state4.y_Start1 = posY
        state4.radius = 0.75
        state4.Ref = "q" & index
        If statetype.Equals("final") Then
            state4.color = Color.Blue
        End If
        m_event = New myEvent
        m_event.Command = "DrawState"
        m_event.m_State = state4
        m_event.MustDo = True
        Commands.Add(m_event)
        If direction.Equals("left") Then
            posX = posX - 4
        Else
            posX = posX + 4
        End If
        index = index + 1
        numstate = numstate + 1
    End Sub

    Private Sub NFAUnion(ByVal direction As
String)
        If Not statetype.Equals("start") Then
            Dim mevent = New myEvent
            If direction.Equals("left") Then
                mevent.m_ArrowDir = "left"
                mevent.m_Arrow = "q" & index &
"to" & "q" & (index - 1)
            Else
                mevent.m_Arrow = "q" & (index -
1) & "to" & "q" & index
            End If
            If down Then
                mevent.Command =
"DrawObscureArrow"
                mevent.m_Arrow = "q" & (index -
1) & "to" & "q" & index
                down = False
            Else
                mevent.Command = "DrawLine"
            End If


            mevent.MustDo = True
            arrayOfArrows.Add(mevent)
            arrayofInputs.Add("ε")
        End If

        Dim m_event As myEvent
        Dim state1 As New State
        state1.x_Start1 = posX
        state1.y_Start1 = posY
        state1.radius = 0.75
        state1.Ref = "q" & index
        If statetype.Equals("start") Then
            state1.color = Color.Firebrick
        End If
        m_event = New myEvent
        m_event.Command = "DrawState"
        m_event.m_State = state1
        m_event.MustDo = True
        Commands.Add(m_event)
        If direction.Equals("left") Then
            posX = posX - 4
        Else
            posX = posX + 4
        End If

        index = index + 1
        numstate = numstate + 1

        m_event = New myEvent
        m_event.Command = "DrawLine"
        If direction.Equals("left") Then
            m_event.m_ArrowDir = "left"
            m_event.m_Arrow = "q" & index &
"to" & "q" & (index - 1)
        Else
            m_event.m_Arrow = "q" & (index - 1)
& "to" & "q" & index
        End If

        m_event.MustDo = True
        arrayOfArrows.Add(m_event)
        arrayofInputs.Add("ε")
```

```
        Dim state2 As New State
        state2.x_Start1 = posX
        state2.y_Start1 = posY + 2
        state2.radius = 0.75
        state2.Ref = "q" & index
        m_event = New myEvent
        m_event.Command = "DrawState"
        m_event.m_State = state2
        m_event.MustDo = True
        Commands.Add(m_event)

        index = index + 1
        numstate = numstate + 1

        m_event = New myEvent
        m_event.Command = "DrawLine"
        If direction.Equals("left") Then
            m_event.m_ArrowDir = "left"
            m_event.m_Arrow = "q" & index &
"to" & "q" & (index - 2)
        Else
            m_event.m_Arrow = "q" & (index - 2)
& "to" & "q" & index
        End If

        m_event.MustDo = True
        arrayOfArrows.Add(m_event)
        arrayofInputs.Add("ε")


        Dim state3 As New State
        state3.x_Start1 = posX
        state3.y_Start1 = posY - 2
        state3.radius = 0.75
        state3.Ref = "q" & index
        m_event = New myEvent
        m_event.Command = "DrawState"
        m_event.m_State = state3
        m_event.MustDo = True
        Commands.Add(m_event)
        If direction.Equals("left") Then
            posX = posX - 4
        Else
            posX = posX + 4
        End If

        index = index + 1
        numstate = numstate + 1

        m_event = New myEvent
        m_event.Command = "DrawLine"
        If direction.Equals("left") Then
            m_event.m_ArrowDir = "left"
            m_event.m_Arrow = "q" & index &
"to" & "q" & (index - 2)
        Else
            m_event.m_Arrow = "q" & (index - 2)
& "to" & "q" & index
        End If

        m_event.MustDo = True
        arrayOfArrows.Add(m_event)
        arrayofInputs.Add("0")


        Dim state4 As New State
        state4.x_Start1 = posX
        state4.y_Start1 = posY + 2
        state4.radius = 0.75
        state4.Ref = "q" & index
        m_event = New myEvent
        m_event.Command = "DrawState"
        m_event.m_State = state4
        m_event.MustDo = True
        Commands.Add(m_event)

        index = index + 1
        numstate = numstate + 1
        m_event = New myEvent
        m_event.Command = "DrawLine"
        If direction.Equals("left") Then
            m_event.m_ArrowDir = "left"
            m_event.m_Arrow = "q" & index &
"to" & "q" & (index - 2)
        Else
            m_event.m_Arrow = "q" & (index - 2)
& "to" & "q" & index
        End If

        m_event.MustDo = True
        arrayOfArrows.Add(m_event)
        arrayofInputs.Add("1")


        Dim state5 As New State
        state5.x_Start1 = posX
        state5.y_Start1 = posY - 2
        state5.radius = 0.75
        state5.Ref = "q" & index
        m_event = New myEvent
        m_event.Command = "DrawState"
        m_event.m_State = state5
        m_event.MustDo = True
        Commands.Add(m_event)
        If direction.Equals("left") Then
            posX = posX - 4
        Else
            posX = posX + 4
        End If

        index = index + 1
        numstate = numstate + 1


        m_event = New myEvent
        m_event.Command = "DrawLine"
        If direction.Equals("left") Then
            m_event.m_ArrowDir = "left"
            m_event.m_Arrow = "q" & index &
"to" & "q" & (index - 1)
        Else
            m_event.m_Arrow = "q" & (index - 1)
& "to" & "q" & index
        End If

        m_event.MustDo = True
        arrayOfArrows.Add(m_event)
        arrayofInputs.Add("ε")

        m_event = New myEvent
        m_event.Command = "DrawLine"
        If direction.Equals("left") Then
            m_event.m_ArrowDir = "left"
            m_event.m_Arrow = "q" & index &
"to" & "q" & (index - 2)
        Else
            m_event.m_Arrow = "q" & (index - 2)
& "to" & "q" & index
        End If

        m_event.MustDo = True
        arrayOfArrows.Add(m_event)
        arrayofInputs.Add("ε")


        Dim state6 As New State
        state6.x_Start1 = posX
        state6.y_Start1 = posY
        state6.radius = 0.75
        state6.Ref = "q" & index
        If statetype.Equals("final") Then
            state6.color = Color.Blue
        End If
        m_event = New myEvent
        m_event.Command = "DrawState"
        m_event.m_State = state6
        m_event.MustDo = True
        Commands.Add(m_event)
        If direction.Equals("left") Then
            posX = posX - 4
        Else
            posX = posX + 4
        End If

        index = index + 1
        numstate = numstate + 1
    End Sub

    Private Sub NFAUnionStar(ByVal direction As
String)
        If Not statetype.Equals("start") Then
            Dim mevent = New myEvent

            If direction.Equals("left") Then
                mevent.m_ArrowDir = "left"
                mevent.m_Arrow = "q" & index &
"to" & "q" & (index - 1)
            Else
                mevent.m_Arrow = "q" & (index -
1) & "to" & "q" & index
            End If
            If down Then
                mevent.Command =
"DrawObscureArrow"
                mevent.m_Arrow = "q" & (index -
1) & "to" & "q" & index
                down = False
            Else
                mevent.Command = "DrawLine"
            End If


            mevent.MustDo = True
            arrayOfArrows.Add(mevent)
            arrayofInputs.Add("ε")
        End If

        Dim m_event As myEvent
        Dim state1 As New State
        state1.x_Start1 = posX
        state1.y_Start1 = posY
        state1.radius = 0.75
        state1.Ref = "q" & index
        If statetype.Equals("start") Then
            state1.color = Color.Firebrick
        End If
        m_event = New myEvent
        m_event.Command = "DrawState"
        m_event.m_State = state1
        m_event.MustDo = True
        Commands.Add(m_event)
        If direction.Equals("left") Then
            posX = posX - 4
        Else
            posX = posX + 4
        End If

        index = index + 1
        numstate = numstate + 1

        m_event = New myEvent
        m_event.Command = "DrawLine"
        If direction.Equals("left") Then
            m_event.m_ArrowDir = "left"
            m_event.m_Arrow = "q" & index &
"to" & "q" & (index - 1)
        Else
            m_event.m_Arrow = "q" & (index - 1)
& "to" & "q" & index
        End If

        m_event.MustDo = True
        arrayOfArrows.Add(m_event)
        arrayofInputs.Add("ε")

        m_event = New myEvent
        m_event.m_obs = 50
        m_event.Command = "DrawObscureArrow"
        If direction.Equals("left") Then
            m_event.m_ArrowDir = "left"
            m_event.m_Arrow = "q" & index - 1 &
"to" & "q" & (index + 6)
        Else
            m_event.m_Arrow = "q" & index - 1 &
"to" & "q" & index + 6
        End If

        m_event.MustDo = True
        arrayOfArrows.Add(m_event)
        arrayofInputs.Add("ε")


        Dim state2 As New State
        state2.x_Start1 = posX
        state2.y_Start1 = posY
        state2.radius = 0.75
        state2.Ref = "q" & index
        m_event = New myEvent
        m_event.Command = "DrawState"
        m_event.m_State = state2
        m_event.MustDo = True
        Commands.Add(m_event)
        If direction.Equals("left") Then
            posX = posX - 4
        Else
            posX = posX + 4
        End If

        index = index + 1
        numstate = numstate + 1

        m_event = New myEvent
        m_event.Command = "DrawLine"
        If direction.Equals("left") Then
            m_event.m_ArrowDir = "left"
            m_event.m_Arrow = "q" & index &
"to" & "q" & (index - 1)
        Else
            m_event.m_Arrow = "q" & (index - 1)
& "to" & "q" & index
        End If

        m_event.MustDo = True
        arrayOfArrows.Add(m_event)
        arrayofInputs.Add("ε")

        m_event = New myEvent
        m_event.m_obs = 40
        m_event.Command = "DrawObscureArrow"
        If direction.Equals("left") Then
            m_event.m_ArrowDir = "left"
            m_event.m_Arrow = "q" & index + 4 &
"to" & "q" & (index - 1)
        Else
            m_event.m_Arrow = "q" & index + 4 &
"to" & "q" & (index - 1)
        End If

        m_event.MustDo = True
        arrayOfArrows.Add(m_event)
        arrayofInputs.Add("ε")

        Dim state3 As New State
        state3.x_Start1 = posX
        state3.y_Start1 = posY + 2
        state3.radius = 0.75
        state3.Ref = "q" & index
        m_event = New myEvent
        m_event.Command = "DrawState"
        m_event.m_State = state3
        m_event.MustDo = True
        Commands.Add(m_event)

        index = index + 1
        numstate = numstate + 1

        m_event = New myEvent
        m_event.Command = "DrawLine"
        If direction.Equals("left") Then
            m_event.m_ArrowDir = "left"
            m_event.m_Arrow = "q" & index &
"to" & "q" & (index - 2)
        Else
            m_event.m_Arrow = "q" & (index - 2)
& "to" & "q" & index
        End If

        m_event.MustDo = True
        arrayOfArrows.Add(m_event)
        arrayofInputs.Add("ε")


        Dim state4 As New State
        state4.x_Start1 = posX
        state4.y_Start1 = posY - 2
        state4.radius = 0.75
        state4.Ref = "q" & index
        m_event = New myEvent
        m_event.Command = "DrawState"
        m_event.m_State = state4
        m_event.MustDo = True
        Commands.Add(m_event)
        If direction.Equals("left") Then
            posX = posX - 4
        Else
            posX = posX + 4
        End If

        index = index + 1
        numstate = numstate + 1



        m_event = New myEvent
        m_event.Command = "DrawLine"
        If direction.Equals("left") Then
            m_event.m_ArrowDir = "left"
            m_event.m_Arrow = "q" & index &
"to" & "q" & (index - 2)
        Else
            m_event.m_Arrow = "q" & (index - 2)
& "to" & "q" & index
        End If

        m_event.MustDo = True
        arrayOfArrows.Add(m_event)
        arrayofInputs.Add("0")


        Dim state5 As New State
        state5.x_Start1 = posX
        state5.y_Start1 = posY + 2
        state5.radius = 0.75
        state5.Ref = "q" & index
        m_event = New myEvent
        m_event.Command = "DrawState"
        m_event.m_State = state5
        m_event.MustDo = True
        Commands.Add(m_event)
        index = index + 1
        numstate = numstate + 1


        m_event = New myEvent
        m_event.Command = "DrawLine"
        If direction.Equals("left") Then
            m_event.m_ArrowDir = "left"
            m_event.m_Arrow = "q" & index &
"to" & "q" & (index - 2)
        Else
            m_event.m_Arrow = "q" & (index - 2)
& "to" & "q" & index
        End If

        m_event.MustDo = True
        arrayOfArrows.Add(m_event)
        arrayofInputs.Add("1")

        Dim state6 As New State
        state6.x_Start1 = posX
        state6.y_Start1 = posY - 2
        state6.radius = 0.75
        state6.Ref = "q" & index
        m_event = New myEvent
        m_event.Command = "DrawState"
        m_event.m_State = state6
        m_event.MustDo = True
        Commands.Add(m_event)
        If direction.Equals("left") Then
            posX = posX - 4
        Else
            posX = posX + 4
        End If


        index = index + 1
        numstate = numstate + 1

        m_event = New myEvent
        m_event.Command = "DrawLine"
        If direction.Equals("left") Then
            m_event.m_ArrowDir = "left"
            m_event.m_Arrow = "q" & index &
"to" & "q" & (index - 1)
        Else
            m_event.m_Arrow = "q" & (index - 1)
& "to" & "q" & index
        End If

        m_event.MustDo = True
        arrayOfArrows.Add(m_event)
        arrayofInputs.Add("ε")

        m_event = New myEvent
        m_event.Command = "DrawLine"
        If direction.Equals("left") Then
            m_event.m_ArrowDir = "left"
            m_event.m_Arrow = "q" & index &
"to" & "q" & (index - 2)
        Else
            m_event.m_Arrow = "q" & (index - 2)
& "to" & "q" & index
        End If

        m_event.MustDo = True
        arrayOfArrows.Add(m_event)
        arrayofInputs.Add("ε")

        Dim state7 As New State
        state7.x_Start1 = posX
        state7.y_Start1 = posY
        state7.radius = 0.75
        state7.Ref = "q" & index
        m_event = New myEvent
        m_event.Command = "DrawState"
        m_event.m_State = state7
        m_event.MustDo = True
        Commands.Add(m_event)
        If direction.Equals("left") Then
            posX = posX - 4
        Else
            posX = posX + 4
        End If

        index = index + 1
        numstate = numstate + 1


        m_event = New myEvent
        m_event.Command = "DrawLine"
        If direction.Equals("left") Then
            m_event.m_ArrowDir = "left"
            m_event.m_Arrow = "q" & index &
"to" & "q" & (index - 1)
        Else
            m_event.m_Arrow = "q" & (index - 1)
& "to" & "q" & index
        End If

        m_event.MustDo = True
        arrayOfArrows.Add(m_event)
        arrayofInputs.Add("ε")

        Dim state8 As New State
        state8.x_Start1 = posX
        state8.y_Start1 = posY
        state8.radius = 0.75
        state8.Ref = "q" & index
        If statetype.Equals("final") Then
            state8.color = Color.Blue
        End If
        m_event = New myEvent
        m_event.Command = "DrawState"
        m_event.m_State = state8
        m_event.MustDo = True
        Commands.Add(m_event)
        If direction.Equals("left") Then
            posX = posX - 4
        Else
            posX = posX + 4
        End If

        index = index + 1
        numstate = numstate + 1

    End Sub

    Private Sub
ComboBox1_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles ComboBox1.SelectedIndexChanged
        setREGEX()
    End Sub

    Private Sub
ComboBox2_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles ComboBox2.SelectedIndexChanged
        setREGEX()
    End Sub

    Private Sub
ComboBox3_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles ComboBox3.SelectedIndexChanged
        setREGEX()
    End Sub

    Private Sub
ComboBox4_SelectedIndexChanged(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles ComboBox4.SelectedIndexChanged
        setREGEX()
    End Sub

    Public Sub setREGEX()
        If Not ComboBox1.SelectedItem Is
Nothing And Not ComboBox2.SelectedItem Is
Nothing And Not ComboBox3.SelectedItem Is
Nothing And Not ComboBox4.SelectedItem Is
Nothing Then
            TextBox1.Text =
ComboBox1.SelectedItem.ToString &
ComboBox2.SelectedItem.ToString &
ComboBox3.SelectedItem.ToString &
ComboBox4.SelectedItem.ToString
        End If
    End Sub

    Private Sub frm_RE_FormClosed(ByVal sender
As System.Object, ByVal e As
System.Windows.Forms.FormClosedEventArgs)
Handles MyBase.FormClosed
        P_parent.Enabled = True
    End Sub

    Private Sub Button2_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button2.Click
        P_parent.Enabled = True
        Me.Close()
    End Sub
End Class
```

Message.designer.vb

```
<Global.Microsoft.VisualBasic.CompilerServices.
DesignerGenerated()> _
Partial Class Message
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the
component list.
    <System.Diagnostics.DebuggerNonUserCode()>
_
    Protected Overrides Sub Dispose(ByVal
disposing As Boolean)
        Try
            If disposing AndAlso components
IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As
System.ComponentModel.IContainer

    'NOTE: The following procedure is required
by the Windows Form Designer
    'It can be modified using the Windows Form
Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()>
_
    Private Sub InitializeComponent()
        Dim resources As
System.ComponentModel.ComponentResourceManager
= New
System.ComponentModel.ComponentResourceManager(
GetType(Message))
        Me.Label1 = New
System.Windows.Forms.Label
        Me.Button1 = New
System.Windows.Forms.Button
        Me.RichTextBox1 = New
System.Windows.Forms.RichTextBox
        Me.Button3 = New
System.Windows.Forms.Button
        Me.RichTextBox2 = New
System.Windows.Forms.RichTextBox
        Me.PictureBox1 = New
System.Windows.Forms.PictureBox
        CType(Me.PictureBox1,
System.ComponentModel.ISupportInitialize).Begin
Init()
        Me.SuspendLayout()
        '
        'Label1
        '
        Me.Label1.AutoSize = True
        Me.Label1.Font = New
System.Drawing.Font("Arial", 9.0!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.Label1.ForeColor =
System.Drawing.Color.Firebrick
        Me.Label1.Location = New
System.Drawing.Point(28, 9)
        Me.Label1.Name = "Label1"
        Me.Label1.Size = New
System.Drawing.Size(86, 15)
        Me.Label1.TabIndex = 1
        Me.Label1.Text = "Error Occured!"
        '
        'Button1
        '
        Me.Button1.Location = New
System.Drawing.Point(31, 128)
        Me.Button1.Name = "Button1"
        Me.Button1.Size = New
System.Drawing.Size(75, 23)
        Me.Button1.TabIndex = 2
        Me.Button1.Text = "Ok"
        Me.Button1.UseVisualStyleBackColor =
True
        '
        'RichTextBox1
        '
        Me.RichTextBox1.BackColor =
System.Drawing.SystemColors.HighlightText
        Me.RichTextBox1.BorderStyle =
System.Windows.Forms.BorderStyle.None
        Me.RichTextBox1.Cursor =
System.Windows.Forms.Cursors.Hand
        Me.RichTextBox1.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Bold,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.RichTextBox1.Location = New
System.Drawing.Point(100, 38)
        Me.RichTextBox1.Name = "RichTextBox1"
        Me.RichTextBox1.ReadOnly = True
        Me.RichTextBox1.Size = New
System.Drawing.Size(198, 20)
        Me.RichTextBox1.TabIndex = 0
        Me.RichTextBox1.Text = "Conversion
Successful!"
        '
        'Button3
        '
        Me.Button3.Cursor =
System.Windows.Forms.Cursors.Hand
        Me.Button3.FlatAppearance.BorderSize =
0
        Me.Button3.FlatAppearance.MouseOverBack
Color =
System.Drawing.SystemColors.HighlightText
        Me.Button3.FlatStyle =
System.Windows.Forms.FlatStyle.Flat
        Me.Button3.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Underline,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.Button3.ForeColor =
System.Drawing.Color.DarkCyan
        Me.Button3.Location = New
System.Drawing.Point(206, 129)
        Me.Button3.Name = "Button3"
        Me.Button3.Size = New
System.Drawing.Size(92, 23)
        Me.Button3.TabIndex = 4
        Me.Button3.Text = "View Details"
        Me.Button3.UseVisualStyleBackColor =
True
        '
        'RichTextBox2
        '
        Me.RichTextBox2.BorderStyle =
System.Windows.Forms.BorderStyle.None
        Me.RichTextBox2.Location = New
System.Drawing.Point(103, 58)
        Me.RichTextBox2.Name = "RichTextBox2"
        Me.RichTextBox2.Size = New
System.Drawing.Size(198, 89)
        Me.RichTextBox2.TabIndex = 5
        Me.RichTextBox2.Text = "          The
Conversion from NFA to DFA is successful. Click
view diagram button" & _
            " to see the transition diagram."
        '
        'PictureBox1
        '
        Me.PictureBox1.Image =
Global.SpecialProject.My.Resources.Resources.x
        Me.PictureBox1.Location = New
System.Drawing.Point(31, 38)
        Me.PictureBox1.Name = "PictureBox1"
        Me.PictureBox1.Size = New
System.Drawing.Size(69, 61)
        Me.PictureBox1.TabIndex = 0
        Me.PictureBox1.TabStop = False
        '
        'Message
        '
        Me.AutoScaleDimensions = New
System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode =
System.Windows.Forms.AutoScaleMode.Font
        Me.BackColor =
System.Drawing.SystemColors.HighlightText
        Me.ClientSize = New
System.Drawing.Size(325, 159)
        Me.Controls.Add(Me.Button3)
        Me.Controls.Add(Me.RichTextBox1)
        Me.Controls.Add(Me.PictureBox1)
        Me.Controls.Add(Me.Button1)
        Me.Controls.Add(Me.Label1)
        Me.Controls.Add(Me.RichTextBox2)
        Me.Icon =
CType(resources.GetObject("$this.Icon"),
System.Drawing.Icon)
        Me.Name = "Message"
        Me.StartPosition =
System.Windows.Forms.FormStartPosition.CenterSc
reen
        Me.Text = "Message"
        CType(Me.PictureBox1,
System.ComponentModel.ISupportInitialize).EndIn
it()
        Me.ResumeLayout(False)
        Me.PerformLayout()

    End Sub
    Friend WithEvents Label1 As
System.Windows.Forms.Label
    Friend WithEvents Button1 As
System.Windows.Forms.Button
    Friend WithEvents RichTextBox1 As
System.Windows.Forms.RichTextBox
    Friend WithEvents PictureBox1 As
System.Windows.Forms.PictureBox
    Friend WithEvents Button3 As
System.Windows.Forms.Button
    Friend WithEvents RichTextBox2 As
System.Windows.Forms.RichTextBox
End Class
```

Message.vb

```
Public Class Message
    Private p_parent As Form

    Public Sub New(ByVal title As String, ByVal
text As String, ByVal type As String, ByVal
message As String, ByVal p As Form)
        ' This call is required by the Windows
Form Designer.
        InitializeComponent()

        ' Add any initialization after the
InitializeComponent() call.
        Me.Focus()
        Me.AcceptButton = Button1
        Me.Size = New Size(341, 188)
        RichTextBox2.Visible = False
        Button1.Location = New Point(31, 114)
        Button3.Location = New Point(206, 64)
        Me.Text = title
        RichTextBox1.Text = text
        If text.Contains("Success") Then
            Label1.ForeColor = Color.DarkGreen
            PictureBox1.Image =
My.Resources.check
        End If
        Label1.Text = type
        RichTextBox2.Text = message
        p_parent = p
    End Sub

    Private Sub Message_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load

    End Sub

    Private Sub Button3_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button3.Click
        If RichTextBox2.Visible Then
            Me.Size = New Size(341, 188)
            RichTextBox2.Visible = False
            Button1.Location = New Point(31,
114)
            Button3.Location = New Point(206,
64)
            Button3.Text = "View Details"
        Else
            Me.Size = New Size(341, 231)
            RichTextBox2.Visible = True
            Button1.Location = New Point(31,
156)
            Button3.Location = New Point(206,
128)
            Button3.Text = "Hide Details"
        End If
    End Sub

    Private Sub Message_FormClosed(ByVal sender
As System.Object, ByVal e As
System.Windows.Forms.FormClosedEventArgs)
Handles MyBase.FormClosed
        p_parent.Enabled = True
    End Sub

    Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
        p_parent.Enabled = True
        Me.Close()
    End Sub
End Class
```

Newform.designer.vb

```
<Global.Microsoft.VisualBasic.CompilerServices.
DesignerGenerated()> _
Partial Class NewForm
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the
component list.
    <System.Diagnostics.DebuggerNonUserCode()>
_
    Protected Overrides Sub Dispose(ByVal
disposing As Boolean)
        Try
            If disposing AndAlso components
IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As
System.ComponentModel.IContainer

    'NOTE: The following procedure is required
by the Windows Form Designer
    'It can be modified using the Windows Form
Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()>
_
    Private Sub InitializeComponent()
        Dim resources As
System.ComponentModel.ComponentResourceManager
= New
System.ComponentModel.ComponentResourceManager(
GetType(NewForm))
        Me.RadioButton1 = New
System.Windows.Forms.RadioButton
        Me.RadioButton2 = New
System.Windows.Forms.RadioButton
        Me.RadioButton3 = New
System.Windows.Forms.RadioButton
        Me.RadioButton4 = New
System.Windows.Forms.RadioButton
        Me.Button1 = New
System.Windows.Forms.Button
        Me.Button2 = New
System.Windows.Forms.Button
        Me.GroupBox1 = New
System.Windows.Forms.GroupBox
        Me.SuspendLayout()
        '
        'RadioButton1
        '
        Me.RadioButton1.AutoSize = True
        Me.RadioButton1.Checked = True
        Me.RadioButton1.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.RadioButton1.Location = New
System.Drawing.Point(32, 29)
        Me.RadioButton1.Name = "RadioButton1"
        Me.RadioButton1.Size = New
System.Drawing.Size(46, 18)
        Me.RadioButton1.TabIndex = 0
        Me.RadioButton1.TabStop = True
        Me.RadioButton1.Text = "DFA"
        Me.RadioButton1.UseVisualStyleBackColor
= True
        '
        'RadioButton2
        '
        Me.RadioButton2.AutoSize = True
        Me.RadioButton2.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.RadioButton2.Location = New
System.Drawing.Point(32, 52)
        Me.RadioButton2.Name = "RadioButton2"
        Me.RadioButton2.Size = New
System.Drawing.Size(46, 18)
        Me.RadioButton2.TabIndex = 1
        Me.RadioButton2.Text = "NFA"
        Me.RadioButton2.UseVisualStyleBackColor
= True
        '
        'RadioButton3
        '
        Me.RadioButton3.AutoSize = True
        Me.RadioButton3.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.RadioButton3.Location = New
System.Drawing.Point(128, 29)
        Me.RadioButton3.Name = "RadioButton3"
        Me.RadioButton3.Size = New
System.Drawing.Size(55, 18)
        Me.RadioButton3.TabIndex = 2
        Me.RadioButton3.TabStop = True
        Me.RadioButton3.Text = "ε-NFA"
        Me.RadioButton3.UseVisualStyleBackColor
= True
        '
        'RadioButton4
        '
        Me.RadioButton4.AutoSize = True
        Me.RadioButton4.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.RadioButton4.Location = New
System.Drawing.Point(128, 52)
        Me.RadioButton4.Name = "RadioButton4"
        Me.RadioButton4.Size = New
System.Drawing.Size(38, 18)
        Me.RadioButton4.TabIndex = 3
        Me.RadioButton4.TabStop = True
        Me.RadioButton4.Text = "RE"
        Me.RadioButton4.UseVisualStyleBackColor
= True
        '
        'Button1
        '
        Me.Button1.Location = New
System.Drawing.Point(32, 91)
        Me.Button1.Name = "Button1"
        Me.Button1.Size = New
System.Drawing.Size(75, 23)
        Me.Button1.TabIndex = 4
        Me.Button1.Text = "Ok"
        Me.Button1.UseVisualStyleBackColor =
True
        '
        'Button2
        '
        Me.Button2.Location = New
System.Drawing.Point(128, 91)
        Me.Button2.Name = "Button2"
        Me.Button2.Size = New
System.Drawing.Size(75, 23)
        Me.Button2.TabIndex = 5
        Me.Button2.Text = "Cancel"
        Me.Button2.UseVisualStyleBackColor =
True
        '
        'GroupBox1
        '
        Me.GroupBox1.Font = New
System.Drawing.Font("Arial", 8.25!,
System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.GroupBox1.Location = New
System.Drawing.Point(17, 12)
        Me.GroupBox1.Name = "GroupBox1"
        Me.GroupBox1.Size = New
System.Drawing.Size(200, 68)
        Me.GroupBox1.TabIndex = 6
        Me.GroupBox1.TabStop = False
        Me.GroupBox1.Text = "Representation"
        '
        'NewForm
        '
        Me.AutoScaleDimensions = New
System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode =
System.Windows.Forms.AutoScaleMode.Font
        Me.BackColor =
System.Drawing.Color.DarkSeaGreen
        Me.ClientSize = New
System.Drawing.Size(229, 126)
        Me.Controls.Add(Me.Button2)
        Me.Controls.Add(Me.Button1)
        Me.Controls.Add(Me.RadioButton4)
        Me.Controls.Add(Me.RadioButton3)
        Me.Controls.Add(Me.RadioButton2)
        Me.Controls.Add(Me.RadioButton1)
        Me.Controls.Add(Me.GroupBox1)
        Me.Icon =
CType(resources.GetObject("$this.Icon"),
System.Drawing.Icon)
        Me.Name = "NewForm"
        Me.Text = "NewForm"
        Me.ResumeLayout(False)
        Me.PerformLayout()

    End Sub
    Friend WithEvents RadioButton1 As
System.Windows.Forms.RadioButton
    Friend WithEvents RadioButton2 As
System.Windows.Forms.RadioButton
    Friend WithEvents RadioButton3 As
System.Windows.Forms.RadioButton
    Friend WithEvents RadioButton4 As
System.Windows.Forms.RadioButton
    Friend WithEvents Button1 As
System.Windows.Forms.Button
    Friend WithEvents Button2 As
System.Windows.Forms.Button
    Friend WithEvents GroupBox1 As
System.Windows.Forms.GroupBox
End Class
```

Newform.vb

```
Public Class NewForm
    Private p_parent As frm_REND
    Private command As String = ""
    Private chose As Boolean = False
    Public Sub New(ByVal f As Form)

        ' This call is required by the Windows
Form Designer.
        InitializeComponent()

        ' Add any initialization after the
InitializeComponent() call.
        p_parent = f
    End Sub

    Private Sub Button2_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button2.Click
        p_parent.Enabled = True
        Me.Close()
    End Sub

    Private Sub NewForm_FormClosed(ByVal sender
As System.Object, ByVal e As
System.Windows.Forms.FormClosedEventArgs)
Handles MyBase.FormClosed
        If Not chose Then
            p_parent.Enabled = True
        End If
    End Sub

    Private Sub Button1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles Button1.Click
        chose = True
        If RadioButton1.Checked Then
            command = "DFA"
        ElseIf RadioButton2.Checked Then
            command = "NFA"
        ElseIf RadioButton3.Checked Then
            command = "NFA-E"
        ElseIf RadioButton4.Checked Then
            command = "RE"
        End If
        p_parent.setNew(command)
        p_parent.Enabled = False
        Me.Close()
    End Sub
    Public ReadOnly Property com()
        Get
            Return command
        End Get
    End Property

    Private Sub NewForm_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load

    End Sub
End Class
```

```
Frm_Reference.designer.vb
```

```
<Global.Microsoft.VisualBasic.CompilerServices.
DesignerGenerated()> _
Partial Class frm_Reference
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the
component list.
    <System.Diagnostics.DebuggerNonUserCode()>
_
    Protected Overrides Sub Dispose(ByVal
disposing As Boolean)
        Try
            If disposing AndAlso components
IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As
System.ComponentModel.IContainer

    'NOTE: The following procedure is required
by the Windows Form Designer
    'It can be modified using the Windows Form
Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()>
_
    Private Sub InitializeComponent()
        Dim TreeNode1 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Types (1)")
        Dim TreeNode2 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Formal
Definition (cont)")
        Dim TreeNode3 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Languages of
DFA")
        Dim TreeNode4 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("DFA as Familiar
Hardware")
        Dim TreeNode5 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Extension of δ
to Paths")
        Dim TreeNode6 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Extended
Transition Function(1)")
        Dim TreeNode7 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Extended
Transition Function(2)")
        Dim TreeNode8 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Acceptance of
Strings")
        Dim TreeNode9 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("State Diagrams")
        Dim TreeNode10 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("State Diagram of
DFA(1)")
        Dim TreeNode11 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("State Diagram of
DFA(2)")
        Dim TreeNode12 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Transition
Table")
        Dim TreeNode13 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("DFA Language")
        Dim TreeNode14 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Heuristic Method
to Determine L(M)")
        Dim TreeNode15 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Heuristic Method
to Design DFAs(1)")
        Dim TreeNode16 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Heuristic Method
to Design DFAs(2)")
        Dim TreeNode17 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Designing DFAs
for Complement Languages")
        Dim TreeNode18 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("DFA’s for
Complement Languages ")
        Dim TreeNode19 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Example")
        Dim TreeNode20 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Incompletely
Specified DFA's")
        Dim TreeNode21 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("State Machine
for 25¢ Vending Machine")
        Dim TreeNode22 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Time Complexity
of DFA's")
        Dim TreeNode23 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Limitations of
DFA's")
        Dim TreeNode24 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Deterministic
Finite Automaton", New
System.Windows.Forms.TreeNode() {TreeNode2,
TreeNode3, TreeNode4, TreeNode5, TreeNode6,
TreeNode7, TreeNode8, TreeNode9, TreeNode10,
TreeNode11, TreeNode12, TreeNode13, TreeNode14,
TreeNode15, TreeNode16, TreeNode17, TreeNode18,
TreeNode19, TreeNode20, TreeNode21, TreeNode22,
TreeNode23})
        Dim TreeNode25 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("NFA Example")
        Dim TreeNode26 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Formal
Definition")
        Dim TreeNode27 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Same Example")
        Dim TreeNode28 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Another
Example")
        Dim TreeNode29 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Why We Study
Non-deterministic Computation?")
        Dim TreeNode30 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Oracle in NFAs")
        Dim TreeNode31 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Non-
deterministic Finite Automaton ", New
System.Windows.Forms.TreeNode() {TreeNode25,
TreeNode26, TreeNode27, TreeNode28, TreeNode29,
TreeNode30})
        Dim TreeNode32 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("ε-Closure")
        Dim TreeNode33 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("ε-Closure
Example")
        Dim TreeNode34 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Formal
Definition")
        Dim TreeNode35 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Non-
deterministic Finite Automaton with epsilon",
New System.Windows.Forms.TreeNode()
{TreeNode32, TreeNode33, TreeNode34})
        Dim TreeNode36 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Types(2)", New
System.Windows.Forms.TreeNode() {TreeNode24,
TreeNode31, TreeNode35})
        Dim TreeNode37 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Application")
        Dim TreeNode38 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Finite
Automaton", New System.Windows.Forms.TreeNode()
{TreeNode1, TreeNode36, TreeNode37})
        Dim TreeNode39 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Properties(1)")
        Dim TreeNode40 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Properties(2)")
        Dim TreeNode41 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Regular
Expression", New
System.Windows.Forms.TreeNode() {TreeNode39,
TreeNode40})
        Dim TreeNode42 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("From NFA to
DFA(1)")
        Dim TreeNode43 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("From NFA to
DFA(2)")
        Dim TreeNode44 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Subset
Construction(1)")
        Dim TreeNode45 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Subset
Construction(2)")
        Dim TreeNode46 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Lazy
Evaluation")
        Dim TreeNode47 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Example")
        Dim TreeNode48 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Conversion from
NFA to DFA", New
System.Windows.Forms.TreeNode() {TreeNode42,
TreeNode43, TreeNode44, TreeNode45, TreeNode46,
TreeNode47})
        Dim TreeNode49 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Subset
Construction")
        Dim TreeNode50 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Example(1)")
        Dim TreeNode51 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Example(2)")
        Dim TreeNode52 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Conversion from
ε - NFA to DFA", New
System.Windows.Forms.TreeNode() {TreeNode49,
TreeNode50, TreeNode51})
        Dim TreeNode53 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("State
Elimination")
        Dim TreeNode54 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("State
Elimination Method")
        Dim TreeNode55 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("State
Elimination Method Rule 1")
        Dim TreeNode56 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("State
Elimination Method Rule 2")
        Dim TreeNode57 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Output")
        Dim TreeNode58 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Hard Case")
        Dim TreeNode59 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Sipser(1)")
        Dim TreeNode60 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Sipser(2)")
        Dim TreeNode61 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Rule")
        Dim TreeNode62 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Step1")
        Dim TreeNode63 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Step2")
        Dim TreeNode64 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Step3")
        Dim TreeNode65 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Final Step")
        Dim TreeNode66 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Conversion from
DFA to RE", New System.Windows.Forms.TreeNode()
{TreeNode53, TreeNode54, TreeNode55,
TreeNode56, TreeNode57, TreeNode58, TreeNode59,
TreeNode60, TreeNode61, TreeNode62, TreeNode63,
TreeNode64, TreeNode65})
        Dim TreeNode67 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Base Case")
        Dim TreeNode68 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Union")
        Dim TreeNode69 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Concatenation")
        Dim TreeNode70 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Kleene Closure")
        Dim TreeNode71 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Conversion from
RE to NFA", New System.Windows.Forms.TreeNode()
{TreeNode67, TreeNode68, TreeNode69,
TreeNode70})
        Dim TreeNode72 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Conversion", New
System.Windows.Forms.TreeNode() {TreeNode48,
TreeNode52, TreeNode66, TreeNode71})
        Dim TreeNode73 As
System.Windows.Forms.TreeNode = New
System.Windows.Forms.TreeNode("Regular
Language", New System.Windows.Forms.TreeNode()
{TreeNode38, TreeNode41, TreeNode72})
        Dim resources As
System.ComponentModel.ComponentResourceManager
= New
System.ComponentModel.ComponentResourceManager(
GetType(frm_Reference))
        Me.LinkLabel1 = New
System.Windows.Forms.LinkLabel
        Me.SaveFileDialog1 = New
System.Windows.Forms.SaveFileDialog
        Me.TreeView1 = New
System.Windows.Forms.TreeView
        Me.MenuStrip1 = New
System.Windows.Forms.MenuStrip
        Me.FileToolStripMenuItem1 = New
System.Windows.Forms.ToolStripMenuItem
        Me.SaveAsToolStripMenuItem1 = New
System.Windows.Forms.ToolStripMenuItem
        Me.ExitToolStripMenuItem1 = New
System.Windows.Forms.ToolStripMenuItem
        Me.FileToolStripMenuItem = New
System.Windows.Forms.ToolStripMenuItem
        Me.SaveAsToolStripMenuItem = New
System.Windows.Forms.ToolStripMenuItem
        Me.ExitToolStripMenuItem = New
System.Windows.Forms.ToolStripMenuItem
        Me.Label1 = New
System.Windows.Forms.Label
        Me.GroupBox1 = New
System.Windows.Forms.GroupBox
        Me.Panel1 = New
System.Windows.Forms.Panel
        Me.PictureBox1 = New
System.Windows.Forms.PictureBox
        Me.Panel2 = New
System.Windows.Forms.Panel
        Me.MenuStrip1.SuspendLayout()
        Me.GroupBox1.SuspendLayout()
        Me.Panel1.SuspendLayout()
        CType(Me.PictureBox1,
System.ComponentModel.ISupportInitialize).Begin
Init()
        Me.SuspendLayout()
        '
        'LinkLabel1
        '
        Me.LinkLabel1.AutoSize = True
        Me.LinkLabel1.Location = New
System.Drawing.Point(1489, 640)
        Me.LinkLabel1.Margin = New
System.Windows.Forms.Padding(4, 0, 4, 0)
        Me.LinkLabel1.Name = "LinkLabel1"
        Me.LinkLabel1.Size = New
System.Drawing.Size(35, 17)
        Me.LinkLabel1.TabIndex = 0
        Me.LinkLabel1.TabStop = True
        Me.LinkLabel1.Text = "DFA"
        '
        'TreeView1
        '
        Me.TreeView1.Location = New
System.Drawing.Point(16, 30)
        Me.TreeView1.Margin = New
System.Windows.Forms.Padding(4)
        Me.TreeView1.Name = "TreeView1"
        TreeNode1.Name = "Node2"
        TreeNode1.Text = "Types (1)"
        TreeNode2.Name = "Node6"
        TreeNode2.Text = "Formal Definition
(cont)"
        TreeNode3.Name = "Node7"
        TreeNode3.Text = "Languages of DFA"
        TreeNode4.Name = "Node8"
        TreeNode4.Text = "DFA as Familiar
Hardware"
        TreeNode5.Name = "Node9"
        TreeNode5.Text = "Extension of δ to
Paths"
        TreeNode6.Name = "Node10"
        TreeNode6.Text = "Extended Transition
Function(1)"
        TreeNode7.Name = "Node11"
        TreeNode7.Text = "Extended Transition
Function(2)"
        TreeNode8.Name = "Node12"
        TreeNode8.Text = "Acceptance of
Strings"
        TreeNode9.Name = "Node13"
        TreeNode9.Text = "State Diagrams"
        TreeNode10.Name = "Node14"
        TreeNode10.Text = "State Diagram of
DFA(1)"
        TreeNode11.Name = "Node15"
        TreeNode11.Text = "State Diagram of
DFA(2)"
        TreeNode12.Name = "Node16"
        TreeNode12.Text = "Transition Table"
        TreeNode13.Name = "Node17"
        TreeNode13.Text = "DFA Language"
        TreeNode14.Name = "Node18"
        TreeNode14.Text = "Heuristic Method to
Determine L(M)"
        TreeNode15.Name = "Node19"
        TreeNode15.Text = "Heuristic Method to
Design DFAs(1)"
        TreeNode16.Name = "Node20"
        TreeNode16.Text = "Heuristic Method to
Design DFAs(2)"
        TreeNode17.Name = "Node21"
        TreeNode17.Text = "Designing DFAs for
Complement Languages"
        TreeNode18.Name = "Node22"
        TreeNode18.Text = "DFA’s for Complement
Languages "
        TreeNode19.Name = "Node23"
        TreeNode19.Text = "Example"
        TreeNode20.Name = "Node24"
        TreeNode20.Text = "Incompletely
Specified DFA's"
        TreeNode21.Name = "Node25"
        TreeNode21.Text = "State Machine for
25¢ Vending Machine"
        TreeNode22.Name = "Node26"
        TreeNode22.Text = "Time Complexity of
DFA's"
        TreeNode23.Name = "Node27"
        TreeNode23.Text = "Limitations of
DFA's"
        TreeNode24.Name = "Node5"
        TreeNode24.Text = "Deterministic Finite
Automaton"
        TreeNode25.Name = "Node29"
        TreeNode25.Text = "NFA Example"
        TreeNode26.Name = "Node30"
        TreeNode26.Text = "Formal Definition"
        TreeNode27.Name = "Node31"
        TreeNode27.Text = "Same Example"
        TreeNode28.Name = "Node32"
        TreeNode28.Text = "Another Example"
        TreeNode29.Name = "Node33"
        TreeNode29.Text = "Why We Study Non-
deterministic Computation?"
        TreeNode30.Name = "Node34"
        TreeNode30.Text = "Oracle in NFAs"
        TreeNode31.Name = "Node28"
        TreeNode31.Text = "Non-deterministic
Finite Automaton "
        TreeNode32.Name = "Node36"
        TreeNode32.Text = "ε-Closure"
        TreeNode33.Name = "Node37"
        TreeNode33.Text = "ε-Closure Example"
        TreeNode34.Name = "Node38"
        TreeNode34.Text = "Formal Definition"
        TreeNode35.Name = "Node35"
        TreeNode35.Text = "Non-deterministic
Finite Automaton with epsilon"
        TreeNode36.Name = "Node3"
        TreeNode36.Text = "Types(2)"
        TreeNode37.Name = "Node4"
        TreeNode37.Text = "Application"
        TreeNode38.Name = "Node1"
        TreeNode38.Text = "Finite Automaton"
        TreeNode39.Name = "Node40"
        TreeNode39.Text = "Properties(1)"
        TreeNode40.Name = "Node41"
        TreeNode40.Text = "Properties(2)"
        TreeNode41.Name = "Node39"
        TreeNode41.Text = "Regular Expression"
        TreeNode42.Name = "Node44"
        TreeNode42.Text = "From NFA to DFA(1)"
        TreeNode43.Name = "Node45"
        TreeNode43.Text = "From NFA to DFA(2)"
        TreeNode44.Name = "Node46"
        TreeNode44.Text = "Subset
Construction(1)"
        TreeNode45.Name = "Node47"
        TreeNode45.Text = "Subset
Construction(2)"
        TreeNode46.Name = "Node48"
        TreeNode46.Text = "Lazy Evaluation"
        TreeNode47.Name = "Node49"
        TreeNode47.Text = "Example"
        TreeNode48.Name = "Node43"
        TreeNode48.Text = "Conversion from NFA
to DFA"
        TreeNode49.Name = "Node51"
        TreeNode49.Text = "Subset Construction"
        TreeNode50.Name = "Node52"
        TreeNode50.Text = "Example(1)"
        TreeNode51.Name = "Node53"
        TreeNode51.Text = "Example(2)"
        TreeNode52.Name = "Node50"
        TreeNode52.Text = "Conversion from ε -
NFA to DFA"
        TreeNode53.Name = "Node55"
        TreeNode53.Text = "State Elimination"
        TreeNode54.Name = "Node56"
        TreeNode54.Text = "State Elimination
Method"
        TreeNode55.Name = "Node57"
        TreeNode55.Text = "State Elimination
Method Rule 1"
        TreeNode56.Name = "Node58"
        TreeNode56.Text = "State Elimination
Method Rule 2"
        TreeNode57.Name = "Node59"
        TreeNode57.Text = "Output"
        TreeNode58.Name = "Node60"
        TreeNode58.Text = "Hard Case"
        TreeNode59.Name = "Node61"
        TreeNode59.Text = "Sipser(1)"
        TreeNode60.Name = "Node62"
        TreeNode60.Text = "Sipser(2)"
        TreeNode61.Name = "Node63"
        TreeNode61.Text = "Rule"
        TreeNode62.Name = "Node64"
        TreeNode62.Text = "Step1"
        TreeNode63.Name = "Node65"
        TreeNode63.Text = "Step2"
        TreeNode64.Name = "Node66"
        TreeNode64.Text = "Step3"
        TreeNode65.Name = "Node67"
        TreeNode65.Text = "Final Step"
        TreeNode66.Name = "Node54"
        TreeNode66.Text = "Conversion from DFA
to RE"
        TreeNode67.Name = "Node69"
        TreeNode67.Text = "Base Case"
        TreeNode68.Name = "Node70"
        TreeNode68.Text = "Union"
        TreeNode69.Name = "Node71"
        TreeNode69.Text = "Concatenation"
        TreeNode70.Name = "Node72"
        TreeNode70.Text = "Kleene Closure"
        TreeNode71.Name = "Node68"
        TreeNode71.Text = "Conversion from RE
to NFA"
        TreeNode72.Name = "Node42"
        TreeNode72.Text = "Conversion"
        TreeNode73.Name = "Node0"
        TreeNode73.Text = "Regular Language"
        Me.TreeView1.Nodes.AddRange(New
System.Windows.Forms.TreeNode() {TreeNode73})
        Me.TreeView1.Size = New
System.Drawing.Size(295, 584)
        Me.TreeView1.TabIndex = 1
        '
        'MenuStrip1
        '
        Me.MenuStrip1.Items.AddRange(New
System.Windows.Forms.ToolStripItem()
{Me.FileToolStripMenuItem1})
        Me.MenuStrip1.Location = New
System.Drawing.Point(0, 0)
        Me.MenuStrip1.Name = "MenuStrip1"
        Me.MenuStrip1.Padding = New
System.Windows.Forms.Padding(8, 2, 0, 2)
        Me.MenuStrip1.Size = New
System.Drawing.Size(1284, 24)
        Me.MenuStrip1.TabIndex = 2
        Me.MenuStrip1.Text = "MenuStrip1"
        '
        'FileToolStripMenuItem1
        '
        Me.FileToolStripMenuItem1.DropDownItems
.AddRange(New
System.Windows.Forms.ToolStripItem()
{Me.SaveAsToolStripMenuItem1,
Me.ExitToolStripMenuItem1})
        Me.FileToolStripMenuItem1.Name =
"FileToolStripMenuItem1"
        Me.FileToolStripMenuItem1.Size = New
System.Drawing.Size(37, 20)
        Me.FileToolStripMenuItem1.Text = "File"
        '
        'SaveAsToolStripMenuItem1
        '
        Me.SaveAsToolStripMenuItem1.Name =
"SaveAsToolStripMenuItem1"
        Me.SaveAsToolStripMenuItem1.Size = New
System.Drawing.Size(123, 22)
        Me.SaveAsToolStripMenuItem1.Text =
"Save As..."
        '
        'ExitToolStripMenuItem1
        '
        Me.ExitToolStripMenuItem1.Name =
"ExitToolStripMenuItem1"
        Me.ExitToolStripMenuItem1.Size = New
System.Drawing.Size(123, 22)
        Me.ExitToolStripMenuItem1.Text = "Exit"
        '
        'FileToolStripMenuItem
        '
        Me.FileToolStripMenuItem.DropDownItems.
AddRange(New
System.Windows.Forms.ToolStripItem()
{Me.SaveAsToolStripMenuItem,
Me.ExitToolStripMenuItem})
        Me.FileToolStripMenuItem.Name =
"FileToolStripMenuItem"
        Me.FileToolStripMenuItem.Size = New
System.Drawing.Size(37, 20)
        Me.FileToolStripMenuItem.Text = "File"
        '
        'SaveAsToolStripMenuItem
        '
        Me.SaveAsToolStripMenuItem.Name =
"SaveAsToolStripMenuItem"
        Me.SaveAsToolStripMenuItem.Size = New
System.Drawing.Size(123, 22)
        Me.SaveAsToolStripMenuItem.Text = "Save
As..."
        '
        'ExitToolStripMenuItem
        '
        Me.ExitToolStripMenuItem.Name =
"ExitToolStripMenuItem"
        Me.ExitToolStripMenuItem.Size = New
System.Drawing.Size(123, 22)
        Me.ExitToolStripMenuItem.Text = "Exit"
        '
        'Label1
        '
        Me.Label1.AutoSize = True
        Me.Label1.Font = New
System.Drawing.Font("Microsoft Sans Serif",
12.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.Label1.ForeColor =
System.Drawing.Color.Ivory
        Me.Label1.Location = New
System.Drawing.Point(7, 0)
        Me.Label1.Margin = New
System.Windows.Forms.Padding(4, 0, 4, 0)
        Me.Label1.Name = "Label1"
        Me.Label1.Size = New
System.Drawing.Size(92, 20)
        Me.Label1.TabIndex = 3
        Me.Label1.Text = "Help Topics"
        '
        'GroupBox1
        '
        Me.GroupBox1.BackColor =
System.Drawing.Color.SeaGreen
        Me.GroupBox1.Controls.Add(Me.Label1)
        Me.GroupBox1.Controls.Add(Me.TreeView1)
        Me.GroupBox1.Location = New
System.Drawing.Point(38, 75)
        Me.GroupBox1.Name = "GroupBox1"
        Me.GroupBox1.Size = New
System.Drawing.Size(329, 633)
        Me.GroupBox1.TabIndex = 4
        Me.GroupBox1.TabStop = False
        Me.GroupBox1.Text = "GroupBox1"
        '
        'Panel1
        '
        Me.Panel1.BackColor =
System.Drawing.Color.Aquamarine
        Me.Panel1.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle
        Me.Panel1.Controls.Add(Me.PictureBox1)
        Me.Panel1.Location = New
System.Drawing.Point(402, 57)
        Me.Panel1.Name = "Panel1"
        Me.Panel1.Size = New
System.Drawing.Size(862, 668)
        Me.Panel1.TabIndex = 5
        '
        'PictureBox1
        '
        Me.PictureBox1.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle
        Me.PictureBox1.Image =
Global.SpecialProject.My.Resources.Resources.In
tro
        Me.PictureBox1.Location = New
System.Drawing.Point(45, 42)
        Me.PictureBox1.Name = "PictureBox1"
        Me.PictureBox1.Size = New
System.Drawing.Size(771, 583)
        Me.PictureBox1.TabIndex = 0
        Me.PictureBox1.TabStop = False
        '
        'Panel2
        '
        Me.Panel2.BackColor =
System.Drawing.Color.SeaGreen
```

```
        Me.Panel2.Location = New
System.Drawing.Point(19, 57)
        Me.Panel2.Name = "Panel2"
        Me.Panel2.Size = New
System.Drawing.Size(364, 668)
        Me.Panel2.TabIndex = 4
        '
        'frm_Reference
        '
        Me.AutoScaleDimensions = New
System.Drawing.SizeF(8.0!, 16.0!)
        Me.AutoScaleMode =
System.Windows.Forms.AutoScaleMode.Font
        Me.BackColor =
System.Drawing.SystemColors.InfoText
        Me.ClientSize = New
System.Drawing.Size(1284, 755)
        Me.Controls.Add(Me.Panel1)
        Me.Controls.Add(Me.LinkLabel1)
        Me.Controls.Add(Me.MenuStrip1)
        Me.Controls.Add(Me.GroupBox1)
        Me.Controls.Add(Me.Panel2)
        Me.Font = New
System.Drawing.Font("Microsoft Sans Serif",
10.0!, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, CType(0,
Byte))
        Me.Icon =
CType(resources.GetObject("$this.Icon"),
System.Drawing.Icon)
        Me.MainMenuStrip = Me.MenuStrip1
        Me.Margin = New
System.Windows.Forms.Padding(4)
        Me.Name = "frm_Reference"
        Me.StartPosition =
System.Windows.Forms.FormStartPosition.CenterSc
reen
        Me.Text = "frm_Reference"
        Me.MenuStrip1.ResumeLayout(False)
        Me.MenuStrip1.PerformLayout()
        Me.GroupBox1.ResumeLayout(False)
        Me.GroupBox1.PerformLayout()
        Me.Panel1.ResumeLayout(False)
        CType(Me.PictureBox1,
System.ComponentModel.ISupportInitialize).EndIn
it()
        Me.ResumeLayout(False)
        Me.PerformLayout()

    End Sub
    Friend WithEvents LinkLabel1 As
System.Windows.Forms.LinkLabel
    Friend WithEvents SaveFileDialog1 As
System.Windows.Forms.SaveFileDialog
    Friend WithEvents TreeView1 As
System.Windows.Forms.TreeView
    Friend WithEvents MenuStrip1 As
System.Windows.Forms.MenuStrip
    Friend WithEvents FileToolStripMenuItem As
System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents SaveAsToolStripMenuItem
As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents ExitToolStripMenuItem As
System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents Label1 As
System.Windows.Forms.Label
    Friend WithEvents GroupBox1 As
System.Windows.Forms.GroupBox
    Friend WithEvents Panel1 As
System.Windows.Forms.Panel
    Friend WithEvents PictureBox1 As
System.Windows.Forms.PictureBox
    Friend WithEvents Panel2 As
System.Windows.Forms.Panel
    Friend WithEvents FileToolStripMenuItem1 As
System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents SaveAsToolStripMenuItem1
As System.Windows.Forms.ToolStripMenuItem
    Friend WithEvents ExitToolStripMenuItem1 As
System.Windows.Forms.ToolStripMenuItem
End Class
```

Frm_Reference.vb

```
Imports System.IO
Public Class frm_Reference
    Public Sub New()

        ' This call is required by the Windows
Form Designer.
        InitializeComponent()

        ' Add any initialization after the
InitializeComponent() call.
    End Sub


    Private Sub TreeView1_AfterSelect(ByVal
sender As System.Object, ByVal e As
System.Windows.Forms.TreeViewEventArgs) Handles
TreeView1.AfterSelect
        Select Case TreeView1.SelectedNode.Name
            Case Is = "Node0"
                PictureBox1.Image =
My.Resources.slide0
            Case Is = "Node1"
                PictureBox1.Image =
My.Resources.slide1
            Case Is = "Node2"
                PictureBox1.Image =
My.Resources.slide2
            Case Is = "Node3"
                PictureBox1.Image =
My.Resources.slide3
            Case Is = "Node4"
                PictureBox1.Image =
My.Resources.slide4
            Case Is = "Node5"
                PictureBox1.Image =
My.Resources.slide5
            Case Is = "Node6"
                PictureBox1.Image =
My.Resources.slide6
            Case Is = "Node7"
                PictureBox1.Image =
My.Resources.slide7
            Case Is = "Node8"
                PictureBox1.Image =
My.Resources.slide8
            Case Is = "Node9"
                PictureBox1.Image =
My.Resources.slide9
            Case Is = "Node10"
                PictureBox1.Image =
My.Resources.slide10
            Case Is = "Node11"
                PictureBox1.Image =
My.Resources.slide11
            Case Is = "Node12"
                PictureBox1.Image =
My.Resources.slide12
            Case Is = "Node13"
                PictureBox1.Image =
My.Resources.slide13
            Case Is = "Node14"
                PictureBox1.Image =
My.Resources.slide14
            Case Is = "Node15"
                PictureBox1.Image =
My.Resources.slide15
            Case Is = "Node16"
                PictureBox1.Image =
My.Resources.slide16
            Case Is = "Node17"
                PictureBox1.Image =
My.Resources.slide17
            Case Is = "Node18"
                PictureBox1.Image =
My.Resources.slide19
```

```
            Case Is = "Node20"
                PictureBox1.Image =
My.Resources.slide20
            Case Is = "Node21"
                PictureBox1.Image =
My.Resources.slide21
            Case Is = "Node22"
                PictureBox1.Image =
My.Resources.slide22
            Case Is = "Node23"
                PictureBox1.Image =
My.Resources.slide23
            Case Is = "Node24"
                PictureBox1.Image =
My.Resources.slide24
            Case Is = "Node25"
                PictureBox1.Image =
My.Resources.slide25
            Case Is = "Node26"
                PictureBox1.Image =
My.Resources.slide26
            Case Is = "Node27"
                PictureBox1.Image =
My.Resources.slide27
            Case Is = "Node28"
                PictureBox1.Image =
My.Resources.slide28
            Case Is = "Node30"
                PictureBox1.Image =
My.Resources.slide30
            Case Is = "Node31"
                PictureBox1.Image =
My.Resources.slide31
            Case Is = "Node32"
                PictureBox1.Image =
My.Resources.slide32
            Case Is = "Node33"
                PictureBox1.Image =
My.Resources.slide33
            Case Is = "Node34"
                PictureBox1.Image =
My.Resources.slide34
            Case Is = "Node35"
                PictureBox1.Image =
My.Resources.slide35
            Case Is = "Node36"
                PictureBox1.Image =
My.Resources.slide36
            Case Is = "Node37"
                PictureBox1.Image =
My.Resources.slide37
            Case Is = "Node38"
                PictureBox1.Image =
My.Resources.slide38
            Case Is = "Node39"
                PictureBox1.Image =
My.Resources.slide39
            Case Is = "Node40"
                PictureBox1.Image =
My.Resources.slide40
            Case Is = "Node41"
                PictureBox1.Image =
My.Resources.slide41
            Case Is = "Node42"
                PictureBox1.Image =
My.Resources.slide42
            Case Is = "Node43"
                PictureBox1.Image =
My.Resources.slide43
            Case Is = "Node44"
                PictureBox1.Image =
My.Resources.slide44
            Case Is = "Node45"
                PictureBox1.Image =
My.Resources.slide45
            Case Is = "Node46"
                PictureBox1.Image =
My.Resources.slide46
            Case Is = "Node47"
                PictureBox1.Image =
My.Resources.slide47
            Case Is = "Node48"
                PictureBox1.Image =
My.Resources.slide48
            Case Is = "Node49"
                PictureBox1.Image =
My.Resources.slide49
            Case Is = "Node50"
                PictureBox1.Image =
My.Resources.slide50
            Case Is = "Node51"
                PictureBox1.Image =
My.Resources.slide51
            Case Is = "Node52"
                PictureBox1.Image =
My.Resources.slide52
            Case Is = "Node53"
                PictureBox1.Image =
My.Resources.slide54
            Case Is = "Node55"
                PictureBox1.Image =
My.Resources.slide55
            Case Is = "Node56"
                PictureBox1.Image =
My.Resources.slide56
            Case Is = "Node57"
                PictureBox1.Image =
My.Resources.slide57
            Case Is = "Node58"
                PictureBox1.Image =
My.Resources.slide58
            Case Is = "Node59"
                PictureBox1.Image =
My.Resources.slide59
            Case Is = "Node60"
                PictureBox1.Image =
My.Resources.slide60
            Case Is = "Node61"
                PictureBox1.Image =
My.Resources.slide61
            Case Is = "Node62"
                PictureBox1.Image =
My.Resources.slide62
            Case Is = "Node63"
                PictureBox1.Image =
My.Resources.slide63
            Case Is = "Node64"
                PictureBox1.Image =
My.Resources.slide64
            Case Is = "Node65"
                PictureBox1.Image =
My.Resources.slide65
            Case Is = "Node66"
                PictureBox1.Image =
My.Resources.slide66
            Case Is = "Node67"
                PictureBox1.Image =
My.Resources.slide67
            Case Is = "Node68"
                PictureBox1.Image =
My.Resources.slide68
            Case Is = "Node69"
                PictureBox1.Image =
My.Resources.slide69
            Case Is = "Node70"
                PictureBox1.Image =
My.Resources.slide70
            Case Is = "Node71"
                PictureBox1.Image =
My.Resources.slide71
            Case Is = "Node72"
                PictureBox1.Image =
My.Resources.slide72
        End Select
    End Sub

    Private Sub
SaveAsToolStripMenuItem1_Click(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles SaveAsToolStripMenuItem1.Click
```

```
        SaveFileDialog1.FileName = "REND-
Reference"
        SaveFileDialog1.Filter =
"Powerpoint(*.ppt)|*.ppt"
        SaveFileDialog1.Title = "Save
PowerPoint Presentation"
        SaveFileDialog1.OverwritePrompt = True

        ' If the file name is not an empty
string open it for saving.
        If SaveFileDialog1.ShowDialog =
DialogResult.OK Then
            System.IO.File.WriteAllBytes(SaveFi
leDialog1.FileName,
My.Resources.REND_Reference)
        End If

    End Sub

    Private Sub frm_Reference_Load(ByVal sender
As System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load

    End Sub
End Class
```

Manual.designer.vb

```
<Global.Microsoft.VisualBasic.CompilerServices.
DesignerGenerated()> _
Partial Class Manual
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the
component list.
    <System.Diagnostics.DebuggerNonUserCode()>
_
    Protected Overrides Sub Dispose(ByVal
disposing As Boolean)
        Try
            If disposing AndAlso components
IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As
System.ComponentModel.IContainer

    'NOTE: The following procedure is required
by the Windows Form Designer
    'It can be modified using the Windows Form
Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()>
_
    Private Sub InitializeComponent()
        Dim resources As
System.ComponentModel.ComponentResourceManager
= New
System.ComponentModel.ComponentResourceManager(
GetType(Manual))
        Me.HelpProvider1 = New
System.Windows.Forms.HelpProvider
        Me.WebBrowser1 = New
System.Windows.Forms.WebBrowser
        Me.SuspendLayout()
        '
        'HelpProvider1
        '
        Me.HelpProvider1.HelpNamespace =
"C:\Users\Essel Tolosa\Desktop\HelpFile.html"
        '
        'WebBrowser1
        '
        Me.WebBrowser1.Dock =
System.Windows.Forms.DockStyle.Fill
        Me.WebBrowser1.Location = New
System.Drawing.Point(0, 0)
        Me.WebBrowser1.MinimumSize = New
System.Drawing.Size(20, 20)
        Me.WebBrowser1.Name = "WebBrowser1"
        Me.WebBrowser1.Size = New
System.Drawing.Size(703, 600)
        Me.WebBrowser1.TabIndex = 0
        '
        'Manual
        '
        Me.AutoScaleDimensions = New
System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode =
System.Windows.Forms.AutoScaleMode.Font
        Me.ClientSize = New
System.Drawing.Size(703, 600)
        Me.Controls.Add(Me.WebBrowser1)
        Me.Icon =
CType(resources.GetObject("$this.Icon"),
System.Drawing.Icon)
        Me.Name = "Manual"
        Me.Text = "Manual"
        Me.ResumeLayout(False)

    End Sub
    Friend WithEvents HelpProvider1 As
System.Windows.Forms.HelpProvider
    Friend WithEvents WebBrowser1 As
System.Windows.Forms.WebBrowser
End Class
```

Manual.vb

```
Imports System
Imports System.IO

Public Class Manual

    Private Sub Manual_Load(ByVal sender As
System.Object, ByVal e As System.EventArgs)
Handles MyBase.Load
        WebBrowser1.Navigate(My.Computer.FileSy
stem.CurrentDirectory & "/HelpFile.html")
    End Sub

    Private Sub
WebBrowser1_DocumentCompleted(ByVal sender As
System.Object, ByVal e As
System.Windows.Forms.WebBrowserDocumentComplete
dEventArgs) Handles
WebBrowser1.DocumentCompleted

    End Sub
End Class
```

About.designer.vb

```
<Global.Microsoft.VisualBasic.CompilerServices.
DesignerGenerated()> _
Partial Class About
    Inherits System.Windows.Forms.Form

    'Form overrides dispose to clean up the
component list.
    <System.Diagnostics.DebuggerNonUserCode()>
_
    Protected Overrides Sub Dispose(ByVal
disposing As Boolean)
        Try
            If disposing AndAlso components
IsNot Nothing Then
                components.Dispose()
            End If
        Finally
            MyBase.Dispose(disposing)
        End Try
    End Sub

    'Required by the Windows Form Designer
    Private components As
System.ComponentModel.IContainer

    'NOTE: The following procedure is required
by the Windows Form Designer
    'It can be modified using the Windows Form
Designer.
    'Do not modify it using the code editor.
    <System.Diagnostics.DebuggerStepThrough()>
_
    Private Sub InitializeComponent()
        Dim resources As
System.ComponentModel.ComponentResourceManager
= New
System.ComponentModel.ComponentResourceManager(
GetType(About))
        Me.PictureBox2 = New
System.Windows.Forms.PictureBox
        Me.Button1 = New
System.Windows.Forms.Button
        CType(Me.PictureBox2,
System.ComponentModel.ISupportInitialize).Begin
Init()
        Me.SuspendLayout()
        '
        'PictureBox2
        '
        Me.PictureBox2.Image =
CType(resources.GetObject("PictureBox2.Image"),
System.Drawing.Image)
        Me.PictureBox2.Location = New
System.Drawing.Point(68, 26)
        Me.PictureBox2.Name = "PictureBox2"
        Me.PictureBox2.Size = New
System.Drawing.Size(308, 207)
        Me.PictureBox2.TabIndex = 8
        Me.PictureBox2.TabStop = False
        '
        'Button1
        '
        Me.Button1.Location = New
System.Drawing.Point(173, 254)
        Me.Button1.Name = "Button1"
        Me.Button1.Size = New
System.Drawing.Size(75, 23)
        Me.Button1.TabIndex = 9
        Me.Button1.Text = "Exit"
        Me.Button1.UseVisualStyleBackColor =
True
        '
        'About
        '
        Me.AutoScaleDimensions = New
System.Drawing.SizeF(6.0!, 13.0!)
        Me.AutoScaleMode =
System.Windows.Forms.AutoScaleMode.Font
        Me.ClientSize = New
System.Drawing.Size(447, 301)
        Me.Controls.Add(Me.Button1)
        Me.Controls.Add(Me.PictureBox2)
        Me.Icon =
CType(resources.GetObject("$this.Icon"),
System.Drawing.Icon)
        Me.Name = "About"
        Me.Text = "About"
        CType(Me.PictureBox2,
System.ComponentModel.ISupportInitialize).EndIn
it()
        Me.ResumeLayout(False)

    End Sub
    Friend WithEvents PictureBox2 As
System.Windows.Forms.PictureBox
        Friend WithEvents Button1 As
    System.Windows.Forms.Button
    End Class
```

About.vb

```
    Public Class About

        Private Sub Button1_Click(ByVal sender As
    System.Object, ByVal e As System.EventArgs)
    Handles Button1.Click
            Me.Close()
        End Sub

        Private Sub About_Load(ByVal sender As
    System.Object, ByVal e As System.EventArgs)
    Handles MyBase.Load

        End Sub
End Class
```

Arrow.vb

```
    Public Class Arrow
        Public state1 As State
        Public state2 As State
End Class
```

Datgridcolumns.vb

```
    Imports System.Text
    Public Class dataGridColumns

        Private P_DataGridCol As
    DataGridViewComboBoxColumn
        Private P_numStates As Integer = 0
        Private P_Header As String = Nothing
        Public P_type As String = Nothing


        Public Sub New(ByVal type As String)
            P_type = type
            P_DataGridCol = New
    DataGridViewComboBoxColumn
            P_DataGridCol.HeaderCell.Style.BackColo
    r = Color.DarkSeaGreen
            P_DataGridCol.DisplayStyle =
    DataGridViewComboBoxDisplayStyle.ComboBox

        End Sub

        Public Sub setNumState(ByVal numStateData
    As Integer)
            P_numStates = numStateData
        End Sub

        Public Sub setHeader(ByVal input As String)
            P_Header = input
            P_DataGridCol.HeaderCell.Value =
    P_Header
        End Sub

        Public ReadOnly Property
    getDataGridColumn()
            Get
                Return Me.P_DataGridCol
            End Get
        End Property


        Public Sub setStartFinal()
            P_DataGridCol.Items.Add("Start")
            P_DataGridCol.Items.Add("Normal")
            P_DataGridCol.Items.Add("Final")
            P_DataGridCol.Items.Add("Start&Final")
    End Sub
    Public Sub setMembers()
        Dim i As Integer
        Dim temp As New StringBuilder
        temp.Append("q")
        Me.P_DataGridCol.Items.Add("ø")
        For i = 0 To P_numStates - 1
            temp.Append(i)
            Me.P_DataGridCol.Items.Add(temp.ToS
tring)
            temp.Replace(i, "")
        Next i

        If P_type.Equals("NFA-E") Or
P_type.Equals("NFA") Then
            If P_numStates > 1 Then
                combineTwo()
            End If
            If P_numStates > 2 Then
                combineThree()
            End If
            If P_numStates > 3 Then
                combineFour()
            End If
            If P_numStates > 4 Then
                combineFive()
            End If
            If P_numStates > 5 Then
                combineSix()
            End If
            If P_numStates > 6 Then
                combineSeven()
            End If
            If P_numStates > 7 Then
                combineEight()
            End If
            If P_numStates > 8 Then
                combineNine()
            End If
            If P_numStates > 9 Then
                combineTen()
            End If
        End If

    End Sub

    Public Sub combineTwo()
        Dim temp As New StringBuilder
        temp.Append("{q")
        For i As Integer = 0 To P_numStates - 1
            For j As Integer = i + 1 To
P_numStates - 1
                temp.Append(i)
                temp.Append(",q")
                temp.Append(j & "}")
                Me.P_DataGridCol.Items.Add(temp
.ToString)
                temp.Replace(i & ",q" & j &
"}", "")
            Next
        Next
    End Sub

    Public Sub combineThree()
        Dim temp As New StringBuilder
        temp.Append("{q")
        For i As Integer = 0 To P_numStates - 1
            For j As Integer = i + 1 To
P_numStates - 1
                For k As Integer = j + 1 To
P_numStates - 1
                    temp.Append(i) :
temp.Append(",q")
                    temp.Append(j) :
temp.Append(",q")
                    temp.Append(k & "}")
                    Me.P_DataGridCol.Items.Add(
temp.ToString)
                    temp.Replace(i & ",q" & j &
",q" & k & "}", "")
                Next
            Next
        Next
    End Sub

    Public Sub combineFour()
        Dim temp As New StringBuilder
        temp.Append("{q")
        For i As Integer = 0 To P_numStates - 1
            For j As Integer = i + 1 To
P_numStates - 1
                For k As Integer = j + 1 To
P_numStates - 1
                    For l As Integer = k + 1 To
P_numStates - 1
                        temp.Append(i) :
temp.Append(",q")
                        temp.Append(j) :
temp.Append(",q")
                        temp.Append(k) :
temp.Append(",q")
                        temp.Append(l & "}")
                        Me.P_DataGridCol.Items.
Add(temp.ToString)
                        temp.Replace(i & ",q" &
j & ",q" & k & ",q" & l & "}", "")
                    Next
                Next
            Next
        Next
    End Sub

    Public Sub combineFive()
        Dim temp As New StringBuilder
        temp.Append("{q")
        For i As Integer = 0 To P_numStates - 1
            For j As Integer = i + 1 To
P_numStates - 1
                For k As Integer = j + 1 To
P_numStates - 1
                    For l As Integer = k + 1 To
P_numStates - 1
                        For m As Integer = l +
1 To P_numStates - 1
                            temp.Append(i) :
temp.Append(",q")
                            temp.Append(j) :
temp.Append(",q")
                            temp.Append(k) :
temp.Append(",q")
                            temp.Append(l) :
temp.Append(",q")
                            temp.Append(m &
"}")
                            Me.P_DataGridCol.It
ems.Add(temp.ToString)
                            temp.Replace(i &
",q" & j & ",q" & k & ",q" & l & ",q" & m &
"}", "")
                        Next
                    Next
                Next
            Next
        Next
    End Sub

    Public Sub combineSix()
        Dim temp As New StringBuilder
        temp.Append("{q")
        For i As Integer = 0 To P_numStates - 1
            For j As Integer = i + 1 To
P_numStates - 1
                For k As Integer = j + 1 To
P_numStates - 1
                    For l As Integer = k + 1 To
P_numStates - 1
                        For m As Integer = l +
1 To P_numStates - 1
                            For n As Integer =
m + 1 To P_numStates - 1
                                temp.Append(i)
: temp.Append(",q")
```

```
                                temp.Append(j)
: temp.Append(",q")
                                temp.Append(k)
: temp.Append(",q")
                                temp.Append(l)
: temp.Append(",q")
                                temp.Append(m)
: temp.Append(",q")
                                temp.Append(n &
"}")
                                Me.P_DataGridCo
l.Items.Add(temp.ToString)
                                temp.Replace(i
& ",q" & j & ",q" & k & ",q" & l & ",q" & m &
",q" & n & "}", "")
                            Next
                        Next
                    Next
                Next
            Next
        Next
    End Sub

    Public Sub combineSeven()
        Dim temp As New StringBuilder
        temp.Append("{q")
        For i As Integer = 0 To P_numStates - 1
            For j As Integer = i + 1 To
P_numStates - 1
                For k As Integer = j + 1 To
P_numStates - 1
                    For l As Integer = k + 1 To
P_numStates - 1
                        For m As Integer = l +
1 To P_numStates - 1
                            For n As Integer =
m + 1 To P_numStates - 1
                                For o As
Integer = n + 1 To P_numStates - 1
                                    temp.Append
(i) : temp.Append(",q")
                                    temp.Append
(j) : temp.Append(",q")
                                    temp.Append
(k) : temp.Append(",q")
                                    temp.Append
(l) : temp.Append(",q")
                                    temp.Append
(m) : temp.Append(",q")
                                    temp.Append
(n) : temp.Append(",q")
                                    temp.Append
(o & "}")
                                    Me.P_DataGr
idCol.Items.Add(temp.ToString)
                                    temp.Replac
e(i & ",q" & j & ",q" & k & ",q" & l & ",q" & m
& ",q" & n & ",q" & o & "}", "")
                                Next
                            Next
                        Next
                    Next
                Next
            Next
        Next
    End Sub

    Public Sub combineEight()
        Dim temp As New StringBuilder
        temp.Append("{q")
        For i As Integer = 0 To P_numStates - 1
            For j As Integer = i + 1 To
P_numStates - 1
                For k As Integer = j + 1 To
P_numStates - 1
                    For l As Integer = k + 1 To
P_numStates - 1
                        For m As Integer = l +
1 To P_numStates - 1
                            For n As Integer =
m + 1 To P_numStates - 1
                                For o As
Integer = n + 1 To P_numStates - 1
                                    For p As
Integer = o + 1 To P_numStates - 1
                                        temp.Ap
pend(i) : temp.Append(",q")
                                        temp.Ap
pend(j) : temp.Append(",q")
                                        temp.Ap
pend(k) : temp.Append(",q")
                                        temp.Ap
pend(l) : temp.Append(",q")
                                        temp.Ap
pend(m) : temp.Append(",q")
                                        temp.Ap
pend(n) : temp.Append(",q")
                                        temp.Ap
pend(o) : temp.Append(",q")
                                        temp.Ap
pend(p & "}")
                                        Me.P_Da
taGridCol.Items.Add(temp.ToString)
                                        temp.Re
place(i & ",q" & j & ",q" & k & ",q" & l & ",q"
& m & ",q" & n & ",q" & o & ",q" & p & "}", "")
                                    Next
                                Next
                            Next
                        Next
                    Next
                Next
            Next
        Next
    End Sub

    Public Sub combineNine()
        Dim temp As New StringBuilder
        temp.Append("{q")
        For i As Integer = 0 To P_numStates - 1
            For j As Integer = i + 1 To
P_numStates - 1
                For k As Integer = j + 1 To
P_numStates - 1
                    For l As Integer = k + 1 To
P_numStates - 1
                        For m As Integer = l +
1 To P_numStates - 1
                            For n As Integer =
m + 1 To P_numStates - 1
                                For o As
Integer = n + 1 To P_numStates - 1
                                    For p As
Integer = o + 1 To P_numStates - 1
                                        For q
As Integer = p + 1 To P_numStates - 1
                                            tem
p.Append(i) : temp.Append(",q")
                                            tem
p.Append(j) : temp.Append(",q")
                                            tem
p.Append(k) : temp.Append(",q")
                                            tem
p.Append(l) : temp.Append(",q")
                                            tem
p.Append(m) : temp.Append(",q")
                                            tem
p.Append(n) : temp.Append(",q")
                                            tem
p.Append(o) : temp.Append(",q")
                                            tem
p.Append(p) : temp.Append(",q")
                                            tem
p.Append(q & "}")
                                            Me.
P_DataGridCol.Items.Add(temp.ToString)
                                            tem
p.Replace(i & ",q" & j & ",q" & k & ",q" & l &
",q" & m & ",q" & n & ",q" & o & ",q" & p &
",q" & q & "}", "")
                                        Next
                                    Next
                            Next
                        Next
                    Next
                Next
            Next
        Next
    Next
End Sub
Public Sub combineTen()
    Dim temp As New StringBuilder
    temp.Append("{q")
    For i As Integer = 0 To P_numStates - 1
        For j As Integer = i + 1 To
P_numStates - 1
            For k As Integer = j + 1 To
P_numStates - 1
                For l As Integer = k + 1 To
P_numStates - 1
                    For m As Integer = l +
1 To P_numStates - 1
                        For n As Integer =
m + 1 To P_numStates - 1
                            For o As
Integer = n + 1 To P_numStates - 1
                                For p As
Integer = o + 1 To P_numStates - 1
                                    For q
As Integer = p + 1 To P_numStates - 1
                                        For
r As Integer = q + 1 To P_numStates - 1

temp.Append(i) : temp.Append(",q")

temp.Append(j) : temp.Append(",q")

temp.Append(k) : temp.Append(",q")

temp.Append(l) : temp.Append(",q")

temp.Append(m) : temp.Append(",q")

temp.Append(n) : temp.Append(",q")

temp.Append(o) : temp.Append(",q")

temp.Append(p) : temp.Append(",q")

temp.Append(q) : temp.Append(",q")

temp.Append(r & "}")

Me.P_DataGridCol.Items.Add(temp.ToString)

temp.Replace(i & ",q" & j & ",q" & k & ",q" & l
& ",q" & m & ",q" & n & ",q" & o & ",q" & p &
",q" & q & ",q" & r & "}", "")
                                        Nex
t
                                    Next
                                Next
                            Next
                        Next
                    Next
                Next
            Next
        Next
    Next
End Sub

End Class
```

DFA.vb

```
Public Class DFA
    Private P_data As DataGridView
    Private P_origData As DataGridView =
Nothing
    Private Commands As ArrayList = Nothing
    Private stateTemp As State
    Private statePos As StatePositions
    Private m_Event As New myEvent
    Private m_text As ArrayList
    Private arrInputs As New ArrayList
    Private alphabet As ArrayList = New
ArrayList()
    Private index As Integer = 0
    Public P_numstate As Integer
    Public C_type As String = Nothing
    Private NFACOnvert As Boolean = False
    Private startState As String = Nothing
    Private FinalState As ArrayList = Nothing
    Private StartFinal As String = Nothing
    Private transitions As String = Nothing

    Public Sub New(ByVal data As DataGridView,
ByVal com As String)

        fillAlphabet()
        index = 0
        P_data = data

        If com.Equals("convertNFA") Then
            C_type = "NFA"
        ElseIf com.Equals("convertNFA-E") Then
            C_type = "NFA-E"
        Else
            C_type = "DFA"
        End If
        startState = getStartState("Start")
        FinalState = getFinalState("Final")
        StartFinal =
getStartFinalState("Start&Final")
        arrInputs = New ArrayList
        Commands = New ArrayList

        If com.Contains("convert") Then
            Dim dt As DataTable =
ConvertNFA(data)
            P_origData = New DataGridView
            P_data = New DataGridView
            P_data.EditMode =
DataGridViewEditMode.EditProgrammatically
            P_origData.EditMode =
DataGridViewEditMode.EditProgrammatically
            For j As Integer = 1 To
dt.Columns.Count - 1
                P_data.Columns.Add("col" & j,
dt.Columns(j).ColumnName)
                P_origData.Columns.Add("col" &
j, dt.Columns(j).ColumnName)
            Next
            For i As Integer = 0 To
dt.Rows.Count - 1
                P_data.Rows.Add()
                P_data.Rows(i).HeaderCell.Value
= dt.Rows(i).Item(0).ToString
                P_origData.Rows.Add()
                P_origData.Rows(i).HeaderCell.V
alue = dt.Rows(i).Item(0).ToString
            Next

            For k As Integer = 2 To
dt.Columns.Count - 1
                For l As Integer = 0 To
dt.Rows.Count - 1
                    P_data.Item(k - 1, l).Value
= dt.Rows(l).Item(k).ToString
                    P_origData.Item(k - 1,
l).Value = dt.Rows(l).Item(k).ToString
                Next
            Next
            P_data.EndEdit()
            P_origData.EndEdit()
            NFACOnvert = True
        Else
            P_data.Rows.Add()
        End If

        processData()

    End Sub
    Public Function getStartState(ByVal text As
String) As String
        Dim s As String = "START"
        Try
            For i As Integer = 0 To
P_data.Rows.Count - 1
                If P_data.Item(0,
i).Value.ToString.Equals(text) Then
                    If C_type.Equals("NFA") Or
C_type.Equals("DFA") Then
                        s =
P_data.Rows(i).HeaderCell.Value
                    ElseIf C_type.Equals("NFA-
E") Then
                        s =
getECLOSE(P_data.Rows(i).HeaderCell.Value)
                    End If
                End If
            Next

        Catch ex As Exception
        End Try
        Return s
    End Function

    Public Function getStartFinalState(ByVal
text As String) As String
        Dim s As String = "STARTFINAL"
        Try

            For i As Integer = 0 To
P_data.Rows.Count - 1
                If P_data.Item(0,
i).Value.ToString.Equals(text) Then
                    If C_type.Equals("NFA") Or
C_type.Equals("DFA") Then
                        s =
P_data.Rows(i).HeaderCell.Value
                    ElseIf C_type.Equals("NFA-
E") Then
                        s =
getECLOSE(P_data.Rows(i).HeaderCell.Value)
                    End If
                End If
            Next
        Catch ex As Exception

        End Try
        Return s
    End Function

    Public Function getFinalState(ByVal text As
String) As ArrayList
        Dim a As New ArrayList
        Try
            For i As Integer = 0 To
P_data.Rows.Count - 1
                If P_data.Item(0,
i).Value.ToString.Equals(text) Then
                    a.Add(P_data.Rows(i).Header
Cell.Value)
                End If
            Next
        Catch ex As Exception

        End Try
        Return a
    End Function
    Public Sub fillAlphabet()
        alphabet.Add("A") : alphabet.Add("B") :
alphabet.Add("C") : alphabet.Add("D") :
alphabet.Add("E") : alphabet.Add("F") :
alphabet.Add("G")
        alphabet.Add("H") : alphabet.Add("I") :
alphabet.Add("J") : alphabet.Add("K") :
alphabet.Add("L") : alphabet.Add("M") :
alphabet.Add("N")
        alphabet.Add("O") : alphabet.Add("P") :
alphabet.Add("Q") : alphabet.Add("R") :
alphabet.Add("S") : alphabet.Add("T") :
alphabet.Add("U")
        alphabet.Add("V") : alphabet.Add("W") :
alphabet.Add("X") : alphabet.Add("Y") :
alphabet.Add("Z")
    End Sub

    Private Sub processData()
        Commands = New ArrayList
        statesCommands()
        arrowCommands()
        'textCommands()
        'labelCommands()
    End Sub

    Public Sub statesCommands()
        statePos = New StatePositions
        For i As Integer = 0 To
Me.P_data.Rows.Count - 2
            stateTemp = New State
            statePos.Position(i + 1,
Me.P_data.Rows.Count, "final")
            stateTemp.x_Start1 = statePos.getX
            stateTemp.y_Start1 = statePos.getY
            stateTemp.z_Start1 = statePos.getZ
            stateTemp.color = Color.Green
            P_data.Item(0, i).Value = "Normal"
            If Not P_origData Is Nothing Then
                P_origData.Item(0, i).Value =
"Normal"
            End If

            Dim x As String =
Me.P_data.Rows.Item(i).HeaderCell.Value
            stateTemp.Name =
Me.P_data.Rows.Item(i).HeaderCell.Value

            If NFACOnvert Then
                stateTemp.Ref = alphabet(index)
                index = index + 1
            Else
                stateTemp.Ref = "q" & i
            End If

            If
P_data.Rows(i).HeaderCell.Value.ToString.Equals
(startState) Then
                stateTemp.color =
Color.Firebrick
                P_data.Item(0, i).Value =
"Start"
                If Not P_origData Is Nothing
Then
                    P_origData.Item(0, i).Value
= "Start"
                End If

            End If
            For l As Integer = 0 To
FinalState.Count - 1
                If
P_data.Rows(i).HeaderCell.Value.ToString.Contai
ns(FinalState(l)) Then
                    stateTemp.color =
Color.Blue
                    P_data.Item(0, i).Value =
"Final"
                    If Not P_origData Is
Nothing Then
                        P_origData.Item(0,
i).Value = "Final"
                    End If

                    If C_type.Equals("NFA-E")
And
P_data.Rows(i).HeaderCell.Value.ToString.Equals
(startState) Then
                        stateTemp.color =
Color.OrangeRed
                        P_data.Item(0, i).Value
= "Start&Final"
                        If Not P_origData Is
Nothing Then
```

```
                            P_origData.Item(0,
i).Value = "Start&Final"
                        End If
                        Exit For
                    End If
                End If
            Next
            If
P_data.Rows(i).HeaderCell.Value.ToString.Equals
(StartFinal) Then
                stateTemp.color =
Color.OrangeRed
                P_data.Item(0, i).Value =
"Start&Final"
                If Not P_origData Is Nothing
Then
                    P_origData.Item(0, i).Value
= "Start&Final"
                End If

            End If

            m_Event = New myEvent
            m_Event.Command = "DrawState"
            m_Event.m_State = stateTemp
            m_Event.MustDo = True
            Commands.Add(m_Event)
        Next
        For i As Integer = 0 To
P_data.Rows.Count - 2
            Dim e As myEvent = Commands(i)
            P_data.Rows(i).HeaderCell.Value =
e.m_State.Ref
        Next
        P_numstate = P_data.Rows.Count - 1

        For i As Integer = 1 To
P_data.Columns.Count - 1
            For j As Integer = 0 To
P_data.Rows.Count - 2
                If Not P_data.Item(i,
j).Value.Equals("⌀") Then
                    P_data.Item(i, j).Value =
findValue(P_data.Item(i, j).Value.ToString)
                End If
            Next
        Next


    End Sub

    Public Function findOrigValue(ByVal s As
String) As String
        For i As Integer = 0 To Commands.Count
- 1
            Dim e As myEvent = Commands(i)
            If e.Command = "DrawState" Then
                If e.m_State.Ref = s Then
                    Return e.m_State.Name
                End If
            End If
        Next
        Return Nothing
    End Function


    Public Function checkEachState() As Boolean
        For i As Integer = 0 To
P_data.Rows.Count - 2
            If
P_data.Rows(i).HeaderCell.Value.ToString.Contai
ns("{") Then
                Return True
                Exit For
            End If
        Next
        Return False
    End Function

    Public Function findValue(ByVal itemName As
String) As String
        For i As Integer = 0 To P_numstate - 1
            Dim e As myEvent = Commands(i)
            If e.Command = "DrawState" Then
                If
e.m_State.Name.Equals(itemName) Then
                    Return e.m_State.Ref
                    Exit For
                End If
            End If
        Next
        Return Nothing
    End Function

    Public Sub arrowCommands()
        arrInputs = New ArrayList
        Dim arr As New ArrayList
        Dim com As String
        Dim e As myEvent
        For j As Integer = 0 To
P_data.Rows.Count - 2
            For k As Integer = 1 To
P_data.Columns.Count - 1
                Dim eTemp As myEvent =
Commands(j)
                Dim x As String =
P_data.Item(k, j).Value.ToString
                com = eTemp.m_State.Ref & "to"
& x
                If Not arr.Contains(com) Then
                    If Not com.Contains("⌀")
Then
                        arr.Add(com)
                        arrInputs.Add(P_data.Co
lumns.Item(k).HeaderText)
                    End If
                Else
                    arrInputs(arr.IndexOf(com))
= arrInputs(arr.IndexOf(com)) & " " & "," & " "
& P_data.Columns.Item(k).HeaderText
                End If
            Next
        Next
        For k As Integer = 0 To arr.Count - 1
            e = New myEvent
            e.m_Arrow = arr(k)
            e.Command = "DrawArrow"
            Commands.Add(e)
        Next
    End Sub


    Public Function ConvertNFA(ByVal _table As
DataGridView) As DataTable
        Dim cont As Boolean = True
        Dim seed As Random = New Random()
        Dim x As String
        Dim dg As New DataTable
        Dim col0 As New DataColumn("States")
        Dim col1 As New
DataColumn(P_data.Columns.Item(0).HeaderText)
        Dim col2 As DataColumn
        Dim col3 As DataColumn
        If C_type.Equals("NFA-E") Then
            col2 = New
DataColumn(P_data.Columns.Item(2).HeaderText)
            col3 = New
DataColumn(P_data.Columns.Item(3).HeaderText)
        Else
            col2 = New
DataColumn(P_data.Columns.Item(1).HeaderText)
            col3 = New
DataColumn(P_data.Columns.Item(2).HeaderText)
        End If

        col1.DataType =
System.Type.GetType("System.String")
        col2.DataType =
System.Type.GetType("System.String")
        col3.DataType =
System.Type.GetType("System.String")
```

```
        dg.Columns.Add(col0)
        dg.Columns.Add(col1)
        dg.Columns.Add(col2)
        dg.Columns.Add(col3)

        Dim row As DataRow
        row = dg.NewRow()
        row(0) =
P_data.Rows(0).HeaderCell.Value
        If C_type.Equals("NFA-E") Then
            row(0) =
getECLOSE(P_data.Rows(0).HeaderCell.Value)
        End If

        dg.Rows.Add(row)
        Dim index As Integer = 0
        While cont
            If index > dg.Rows.Count - 1 Then
                Exit While
            End If
            For j As Integer = 2 To
dg.Columns.Count - 1
                x = processItem(index, j, dg,
P_data)
                dg.Rows(index).Item(j) = x
                If Not
dg.Rows(index).Item(j).ToString.Equals("ø") And
Not checkIfRowExist(dg,
dg.Rows(index).Item(j).ToString) Then
                    row = dg.NewRow
                    dg.Rows.Add(row)
                    dg.Rows(dg.Rows.Count -
1).Item(0) = x
                End If
            Next
            index = index + 1
        End While



        Return dg
    End Function

    Public Function processItem(ByVal row As
Integer, ByVal col As Integer, ByVal _table As
DataTable, ByVal data As DataGridView) As
String
        Dim rowName As String
        Dim s As String = Nothing
        Dim a As ArrayList = New ArrayList

        rowName =
_table.Rows(row).Item(0).ToString
        If Not
_table.Rows(row).Item(col).ToString = "ø" Then
            If rowName.Contains("{") Then
                rowName = rowName.Replace("{",
"")
                rowName = rowName.Replace("}",
"")
                rowName = rowName.Replace(",",
" ")
                Dim z As Array =
rowName.Split(" ")
                For i As Integer = 0 To
z.Length - 1
                    a.Add(getValue(col, row,
z(i), data))
                Next
                s = getValue(a)
            Else
                s = getValue(col, row, rowName,
data)
            End If
            If s.Length = 3 Then
                s = s.Replace("{", "")
                s = s.Replace("}", "")
            End If
            If s.Length > 3 Then
                s = s.Replace(",ø", "")
            End If
            If s.Equals("}") Or s.Equals("")
Then
                s = "ø"
            End If
        Else
            s = "ø"
        End If

        If s = "ø" Then
            Return s
            Exit Function
        End If

        Return Arrange(s)
    End Function

    Public Function Arrange(ByVal s As String)
As String
        Dim z As String = s.Replace("{", "")
        z = z.Replace("}", "")
        z = z.Replace(",", " ")
        Dim a As Array = z.Split(" ")
        Array.Sort(a)
        Dim value As String = "{"
        For i As Integer = 0 To a.Length - 2
            value = value & a(i) & ","
        Next
        value = value & a(a.Length - 1) & "}"
        If value.Length = 4 Then
            value = value.Replace("{", "")
            value = value.Replace("}", "")
        End If
        Return value
    End Function


    Public Function getValue(ByVal a As
ArrayList) As String
        Dim value As String = "{"

        For i As Integer = 0 To a.Count - 1
            Dim s As String = a(i)
            If s.Contains("{") Then
                s = s.Replace("{", "")
                s = s.Replace("}", "")
                s = s.Replace(",", " ")
                Dim z As Array = s.Split(" ")
                For j As Integer = 0 To
z.Length - 1
                    If Not value.Contains(z(j))
Then
                        value = value & z(j) &
","
                    End If
                Next
            Else
                If Not value.Contains(s) Then
                    value = value & s & ","
                End If
            End If
        Next
        value = value.Remove(value.Length - 1,
1)
        If Not value.Equals("") Then
            value = value & "}"
        End If

        Return value
    End Function

    Public Function getValue(ByVal col As
Integer, ByVal row As Integer, ByVal rowName As
String, ByVal _table As DataGridView) As String
        For i As Integer = 0 To
_table.Rows.Count - 1
            If P_data.Rows(i).HeaderCell.Value
= rowName Then
                If C_type.Equals("NFA") Then
                    Dim x As String =
P_data.Item(col - 1, i).Value
                    Return x
                    Exit For
                ElseIf C_type.Equals("NFA-E")
Then
                    Dim x As String =
P_data.Item(col, i).Value
                    Return getECLOSE(x)
                End If
            End If
        Next
        Return Nothing
    End Function

    Public Function getECLOSE(ByVal x As
String) As String
        Dim value As String = "{"
        Dim arr As New ArrayList
        If x.Contains("{") Then
            x = x.Replace("{", "")
            x = x.Replace("}", "")
            x = x.Replace(",", " ")
            Dim a As Array = x.Split(" ")
            For i As Integer = 0 To a.Length -
1
                For j As Integer = 0 To
P_data.Rows.Count - 1
                    If
P_data.Rows(j).HeaderCell.Value.Equals(a(i))
Then
                        Dim z As String =
P_data.Item(1, j).Value
                        arr.Add(z)
                    End If
                Next
            Next
        Else
            For j As Integer = 0 To
P_data.Rows.Count - 1
                If
P_data.Rows(j).HeaderCell.Value = x Then
                    Dim z As String =
P_data.Item(1, j).Value
                    Return z
                    Exit For
                End If
            Next
        End If

        Return getValue(arr)
    End Function

    Public Function checkIfRowExist(ByVal
_table As DataTable, ByVal row As String) As
Boolean
        For i As Integer = 0 To
_table.Rows.Count - 1
            If _table.Rows(i).Item(0).ToString
= row Then
                Return True
            End If
        Next
        Return False
    End Function



    Public ReadOnly Property Table() As
DataGridView
        Get
            Return P_data
        End Get
    End Property

    Public ReadOnly Property ConvertTable() As
DataGridView
        Get
            Return P_origData
        End Get
    End Property

    Public ReadOnly Property checkNumbers(ByVal
text As String, ByVal arr As ArrayList) As
Integer
        Get
            Dim count As Integer = 0
            For i As Integer = 0 To arr.Count -
1
                If arr(i).ToString.Equals(text)
Then
                    count = count + 1
                End If
            Next
            Return count
        End Get
    End Property

    Public ReadOnly Property
checkIthPosition(ByVal index As Integer, ByVal
text As String, ByVal arr As ArrayList) As
Integer
        Get
            Dim pos As Integer = 0
            For i As Integer = 0 To index
                If arr(i).ToString.Equals(text)
Then
                    pos = pos + 1
                End If
            Next
            Return pos
        End Get
    End Property

    Public ReadOnly Property getCommands() As
ArrayList
        Get
            If Commands Is Nothing Then
                Return Nothing
            End If
            Return Commands
        End Get
    End Property

    Public ReadOnly Property getInputs() As
ArrayList
        Get
            If arrInputs Is Nothing Then
                Return Nothing
            End If
            Return arrInputs
        End Get
    End Property

End Class
```

NFA.vb

```
Public Class NFA
    Private P_data As DataGridView
    Private Commands As ArrayList = Nothing
    Private stateTemp As State
    Private statePos As StatePositions
    Private m_Event As New myEvent
    Private m_text As ArrayList
    Private arrInputs As New ArrayList

    Public Sub New(ByVal data As DataGridView)
        P_data = data
        Commands = New ArrayList
        arrInputs = New ArrayList
        processData()
    End Sub

    Private Sub processData()
        Commands = New ArrayList
        statesCommands()
        arrowCommands()
        'textCommands()
        'labelCommands()
    End Sub
    Public Sub statesCommands()
        statePos = New StatePositions
        For i As Integer = 0 To
P_data.Rows.Count - 1
            stateTemp = New State
            statePos.Position(i + 1,
P_data.Rows.Count, "final")
            stateTemp.x_Start1 = statePos.getX
            stateTemp.y_Start1 = statePos.getY
            stateTemp.z_Start1 = statePos.getZ
            stateTemp.Name =
P_data.Rows(i).HeaderCell.Value
            stateTemp.Ref = "q" & i
            If P_data.Item(0, i).Value =
"Start" Then
                stateTemp.color =
Color.Firebrick
            End If
            If P_data.Item(0, i).Value =
"Final" Then
                stateTemp.color = Color.Blue
            End If
            If P_data.Item(0, i).Value =
"Start&Final" Then
                stateTemp.color =
Color.OrangeRed
            End If
            m_Event = New myEvent
            m_Event.Command = "DrawState"
            m_Event.m_State = stateTemp
            m_Event.MustDo = True
            Commands.Add(m_Event)
        Next
    End Sub

    Public Sub arrowCommands()
        Dim arr As New ArrayList
        Dim e As myEvent
        Dim com As String
        Dim pos As New ArrowsPosition
        Dim arrow As New Arrow
        For j As Integer = 0 To
P_data.Rows.Count - 1
            For k As Integer = 1 To
P_data.Columns.Count - 1
                If P_data.Item(k,
j).Value.ToString.Contains("{") Then
                    Dim x As String =
P_data.Item(k, j).Value.ToString
                    x = x.Replace("{", "")
                    x = x.Replace("}", "")
                    x = x.Replace(",", " ")
                    Dim z As Array = x.Split("
")
                    For i As Integer = 0 To
z.Length - 1
                        com = "q" & j & "to" &
z(i).ToString
                        If Not
arr.Contains(com) Then
                            If Not
com.Contains("ø") Then
                                arr.Add(com)
                                arrInputs.Add(P
_data.Columns.Item(k).HeaderText)
                            End If
                        Else
                            arrInputs(arr.Index
Of(com)) = arrInputs(arr.IndexOf(com)) & " " &
"," & " " & P_data.Columns.Item(k).HeaderText
                        End If
                    Next
                Else
                    com = "q" & j & "to" &
P_data.Item(k, j).Value.ToString
                    If Not arr.Contains(com)
Then
                        If Not
com.Contains("ø") Then
                            arr.Add(com)
                            arrInputs.Add(P_dat
a.Columns.Item(k).HeaderText)
                        End If
                    Else
                        arrInputs(arr.IndexOf(c
om)) = arrInputs(arr.IndexOf(com)) & " " & ","
& " " & P_data.Columns.Item(k).HeaderText
                    End If
                End If
            Next
        Next

        For k As Integer = 0 To arr.Count - 1
            e = New myEvent
            e.m_Arrow = arr(k)
            e.Command = "DrawArrow"
            Commands.Add(e)
        Next

    End Sub

    Public Sub textCommands()
        Dim arr As New ArrayList
        Dim com As String
        Dim inputs As New ArrayList

        For j As Integer = 0 To
P_data.Rows.Count - 1
            arr = New ArrayList
            inputs = New ArrayList
            For k As Integer = 1 To
P_data.Columns.Count - 1
                If P_data.Item(k,
j).Value.ToString.Contains("{") Then
                    Dim x As String =
P_data.Item(k, j).Value.ToString
                    x = x.Replace("{", "")
                    x = x.Replace("}", "")
                    x = x.Replace(",", " ")
                    Dim z As Array = x.Split("
")
                    For i As Integer = 0 To
z.Length - 1
                        com = "q" & j & "to" &
z(i).ToString
                        arr.Add(com)
                        arrInputs.Add(P_data.Co
lumns(k).HeaderText)
                    Next
                Else
                    com = "q" & j & "to" &
P_data.Item(k, j).Value.ToString
                    arr.Add(com)
                    arrInputs.Add(P_data.Column
s(k).HeaderText)
                End If
            Next
        Next


    End Sub



    Public ReadOnly Property checkNumbers(ByVal
text As String, ByVal arr As ArrayList) As
Integer
        Get
            Dim count As Integer = 0
            For i As Integer = 0 To arr.Count -
1
                If arr(i).ToString.Equals(text)
Then
                    count = count + 1
                End If
            Next
            Return count
        End Get
    End Property
    Public ReadOnly Property
checkIthPosition(ByVal index As Integer, ByVal
text As String, ByVal arr As ArrayList) As
Integer
        Get
            Dim pos As Integer = 0
            For i As Integer = 0 To index
                If arr(i).ToString.Equals(text)
Then
                    pos = pos + 1
                End If
            Next
            Return pos
        End Get
    End Property

    Public ReadOnly Property getCommands() As
ArrayList
        Get
            If Commands Is Nothing Then
                Return Nothing
            End If
            Return Commands
        End Get
    End Property

    Public ReadOnly Property Table() As
DataGridView
        Get
            If P_data Is Nothing Then
                Return Nothing
            End If
            Return P_data
        End Get
    End Property

    Public ReadOnly Property getInputs() As
ArrayList
        Get
            If arrInputs Is Nothing Then
                Return Nothing
            End If
            Return arrInputs
        End Get
    End Property

End Class
```

PointCLass.vb

```
Public Class PointClass
    Public x As Double
    Public y As Double

    Public Sub New(ByVal x As Double, ByVal y
As Double)
        Me.x = x
        Me.y = y
    End Sub
End Class
```

Question.vb

```
Public Class Question
    Private q As ArrayList = Nothing
    Private type As String = Nothing
    Private choice1 As ArrayList = Nothing
    Private choice2 As ArrayList = Nothing
    Private choice3 As ArrayList = Nothing
    Private choice4 As ArrayList = Nothing
    Private answer As ArrayList = Nothing
    Private asked As ArrayList = Nothing
    Private RandomNumber As Integer
    Private RandomCLass As Random
    Public Sub New(ByVal t As String)
        type = t
        RandomCLass = New Random()
        Populate(type)
        asked = New ArrayList
    End Sub

    Public ReadOnly Property getQuestion()
        Get
            RandomNumber = RandomCLass.Next(0,
q.Count - 1)
            If Not asked.Contains(RandomNumber)
Then
                asked.Add(RandomNumber)
            Else
                While
asked.Contains(RandomNumber)
                    RandomNumber =
RandomCLass.Next(0, q.Count - 1)
                End While
                asked.Add(RandomNumber)
            End If
            Dim x As ArrayList = New ArrayList
            x.Add(q(RandomNumber))
            If type.Equals("EASY") Then
                x.Add(choice1(0))
                x.Add(choice2(0))
            ElseIf type.Equals("INTERMEDIATE")
Then
                x.Add(choice1(RandomNumber))
                x.Add(choice2(RandomNumber))
                x.Add(choice3(RandomNumber))
                x.Add(choice4(RandomNumber))

            End If
            x.Add(answer(RandomNumber))
            Return x
        End Get
    End Property

    Public Sub Populate(ByVal t As String)
        Select Case t
            Case Is = "EASY"
                retrieveDataEasy()
            Case Is = "INTERMEDIATE"
                retrieveDataIntermediate()
            Case Is = "HARD"
                retrieveDataHard()
        End Select
    End Sub

    Public Sub retrieveDataEasy()
        q = New ArrayList
        answer = New ArrayList
        choice1 = New ArrayList
        choice2 = New ArrayList
        choice1.Add("TRUE")
        choice2.Add("FALSE")
        q.Add("aab is in the language
(a+b)*(a+bb)")
        answer.Add("False")
        q.Add("abaa is in the language
(a+b)*(a+bb)")
        answer.Add("TRUE")
        q.Add("ε* is empty")
        answer.Add("FALSE")
        q.Add("(a*+b)* = (a+b)*")
        answer.Add("TRUE")
        q.Add("(b + ab*a)* is the set of
strings that contain even number of a")
        answer.Add("TRUE")
        q.Add("abaabbab is in the language
(a+ba)*(b+ε)")
        answer.Add("FALSE")
        q.Add("(r+ε)* = r* for any regular
expression r")
        answer.Add("TRUE")
        q.Add("b*ab*ab* is the set of strings
that contain exactly two a's")
        answer.Add("TRUE")
        q.Add("(aa)*(ε+a) = a*")
        answer.Add("TRUE")
        q.Add("a(aa)*(ε+a)b + b = a*b")
        answer.Add("TRUE")
        q.Add("If lengths of the string are
multiples of 3, then L is regular")
```

```
        answer.Add("FALSE")
        q.Add("If L over Σ1 is regular then L
over any Σ containing Σ1 is regular")
        answer.Add("TRUE")
        q.Add("In a DFA all states have the
same number of transitions")
        answer.Add("TRUE")
        q.Add("When a is read at q, NFA goes at
a state in δ(q,a)")
        answer.Add("TRUE")
        q.Add("An NFA can modify its input")
        answer.Add("FALSE")
        q.Add("The tape head of an NFA can move
backward as well")
        answer.Add("FALSE")
        q.Add("When the transitions for all the
states in DFA is identified, the DFA will have
been obtained")
        answer.Add("TRUE")
        q.Add("The set consisting of the
initial state of NFA is the initial state of a
DFA")
        answer.Add("TRUE")
        q.Add("In the union of FAs all the
accepting states can be coaleseced to form one
accepting state")
        answer.Add("FALSE")
        q.Add("The intersection of language and
its complement is regular")
        answer.Add("TRUE")
    End Sub
    Public Sub retrieveDataIntermediate()
        q = New ArrayList
        answer = New ArrayList
        choice1 = New ArrayList
        choice2 = New ArrayList
        choice3 = New ArrayList
        choice4 = New ArrayList
        q.Add("Does not accept a regular
language")
        choice1.Add("DFA")
        choice2.Add("NFA")
        choice3.Add("Both")
        choice4.Add("None")
        answer.Add("None")
        q.Add("Abstract machine that processes
a string of symbols and decides whether to
accept or reject the string")
        choice1.Add("DFA")
        choice2.Add("Finite Automaton")
        choice3.Add("NFA")
        choice4.Add("Regular Expression")
        answer.Add("Finite Automaton")
        q.Add("Decreases the burden of
performing transitions of each state in the
process of converting NFA to DFA")
        choice1.Add("Lazy Evaluation")
        choice2.Add("Subset Construction")
        choice3.Add("Tape reverse")
        choice4.Add("E-close")
        answer.Add("Lazy Evaluation")
        q.Add("Consists of string that contains
exactly 2 b's")
        choice1.Add("a*ba*b*b")
        choice2.Add("a*bba*b")
        choice3.Add("aaba*aa*b")
        choice4.Add("aaaabb*")
        answer.Add("aaba*aa*b")
        q.Add("This cannot have more than one
state transtion from a unique input")
        choice1.Add("NFA")
        choice2.Add("DFA")
        choice3.Add("RE")
        choice4.Add("All")
        answer.Add("DFA")
        q.Add("Representation of a Regular
Language")
        choice1.Add("DFA")
        choice2.Add("NFA")
        choice3.Add("Regular Expression")
        choice4.Add("All")
        answer.Add("All")
        q.Add("Cannot have more than one
accepting state")
        choice1.Add("DFA")
        choice2.Add("NFA")
        choice3.Add("Regular Expression")
        choice4.Add("None")
        answer.Add("None")
        q.Add("Possible or maximum number of
states from converting NFA with 5 states to
DFA")
        choice1.Add("25")
        choice2.Add("30")
        choice3.Add("32")
        choice4.Add("16")
        answer.Add("32")
        q.Add("Uses memory to store inputs")
        choice1.Add("DFA")
        choice2.Add("NFA")
        choice3.Add("Both")
        choice4.Add("None")
        answer.Add("None")
        q.Add("a* - ε equals?")
        choice1.Add("a?")
        choice2.Add("a*")
        choice3.Add("a+")
        choice4.Add("a")
        answer.Add("a+")
        q.Add("a* - a+ - ε is equal to?")
        choice1.Add("empty")
        choice2.Add("a")
        choice3.Add("ε")
        choice4.Add("b")
        answer.Add("empty")
        q.Add("Any regular language is accepted
by Finite Automaton")
        choice1.Add("Kleene's Theorem")
        choice2.Add("Kleene CLosure")
        choice3.Add("Automata Theory")
        choice4.Add("Acceptance Theorem")
        answer.Add("Kleene's Theorem")
        q.Add("Obtained by swapping accepting
states and start states")
        choice1.Add("Converse")
        choice2.Add("Complement")
        choice3.Add("Inverse")
        choice4.Add("Intersection")
        answer.Add("Complement")
        q.Add("a*b*c* is regular")
        choice1.Add("True")
        choice2.Add("Sometimes True")
        choice3.Add("False")
        choice4.Add("Can't answer")
        answer.Add("False")
    End Sub

    Public Sub retrieveDataHard()
        q = New ArrayList
        answer = New ArrayList
        q.Add("Find the shortest string that is
not in the language represented by the regular
expression a*(ab)*b*.")
        answer.Add("It can easily be seen
that , a, b, which are strings in the language
with length 1 or less. Of the strings wiht
length 2 aa, bb and ab are in the language.
However, ba is not in it. Thus the answer is
ba.")
        q.Add("For the two regular expressions
given below, " & vbCrLf & _
              "(a) find a string corresponding
to r2 but not to r1 and " & vbCrLf & _
              "(b) find a string corresponding
to both r1 and r2." & _
              vbCrLf & vbCrLf & "r1 = a* + b*
r2 = ab* + ba* + b*a + (a*b)* ")
        answer.Add("(a) Any string consisting
of only a's or only b's and the empty string
are in r1. So we need to find strings of r2
which contain at least one a and at least one
b. For example ab and ba are such strings." & _
```

```
                   "(b) A string corresponding
to r1 consists of only a's or only b's or the
empty string. The only strings corresponding to
r2 which consist of only a's or b's are a, b
and the strings consiting of only b's (from
(a*b)*). ")
        q.Add("Let r1 and r2 be arbitrary
regular expressions over some alphabet. Find a
simple (the shortest and with the smallest
nesting of * and +) regular expression which is
equal to each of the following regular
expressions." & vbCrLf & _
               "(a) (r1 + r2 + r1r2 + r2r1)*" &
vbCrLf & "(b) (r1(r1 + r2)*)+)")
        answer.Add("One general strategy to
approach this type of question is to try to see
whether or not they are equal to simple regular
expressions that are familiar to us such as a,
a*, a+, (a + b)*, (a + b)+ etc." & _
"(a) Since (r1 + r2)* represents all strings
consisting of strings of r1 and/or r2 , r1r2 +
r2r1 in the given regular expression is
redundant, that is, they do not produce any
strings that are not represented by (r1 + r2)*.
Thus (r1 + r2 + r1r2 + r2r1)* is reduced to (r1
+ r2)*." & _
"(b) (r1(r1 + r2)*)+ means that all the strings
represented by it must consist of one or more
strings of (r1(r1 + r2)*). However, the strings
of (r1(r1 + r2)*) start with a string of r1
followed by any number of strings taken
arbitrarily from r1 and/or r2. Thus anything
that comes after the first r1 in (r1(r1 + r2)*)
+ is represented by (r1 + r2)*. Hence (r1(r1 +
r2)*) also represents the strings of (r1(r1 +
r2)*)+, and conversely (r1(r1 + r2)*)+
represents the strings represented by (r1(r1 +
r2)*). Hence (r1(r1 + r2)*)+ is reduced to
(r1(r1 + r2)*).")
        q.Add("Find a regular expression
corresponding to the language of all strings
over the alphabet { a, b } that contain exactly
two a's. ")
        answer.Add("A string in this language
must have at least two a's. Since any string of
b's can be placed in front of the first a,
behind the second a and between the two a's,
and since an arbitrasry string of b's can be
represented by the regular expression b*, b*a
b*a b* is a regular expression for this
language. ")
        q.Add("Find a regular expression
corresponding to the language of all strings
over the alphabet { a, b } that do not end with
ab. ")
        answer.Add("Any string in a language
over { a , b } must end in a or b. Hence if a
string does not end with ab then it ends with a
or if it ends with b the last b must be
preceded by a symbol b. Since it can have any
string in front of the last a or bb, ( a + b )
*( a + bb ) is a regular expression for the
language. ")
        q.Add("Find a regular expression
corresponding to the language of strings of
even lengths over the alphabet of { a, b }. ")
        answer.Add("Since any string of even
length can be expressed as the concatenation of
strings of length 2 and since the strings of
length 2 are aa, ab, ba, bb, a regular
expression corresponding to the language is
( aa + ab + ba + bb )*. Note that 0 is an even
number. Hence the string ε is in this
language.")
        q.Add("Describe as simply as possible
in English the language corresponding to the
regular expression a*b(a*ba*b)*a* .")
        answer.Add("A string in the language
can start and end with a or b, it has at least
one b, and after the first b all the b's in the
string appear in pairs. Any numbe of a's can
appear any place in the string. Thus simply
put, it is the set of strings over the alphabet
{ a, b } that contain an odd number of b's ")
        q.Add("Describe as simply as possible
in English the language corresponding to the
regular expression (( a + b )3)*( ε + a +
b ) .")
        answer.Add("(( a + b )3) represents the
strings of length 3. Hence (( a + b )3)*
represents the strings of length a multiple of
3. Since (( a + b )3)*( a + b ) represents the
strings of length 3n + 1, where n is a natural
number, the given regular expression represents
the strings of length 3n and 3n + 1, where n is
a natural number.")
        q.Add("Describe as simply as possible
in English the language corresponding to the
regular expression ( b + ab )*( a + ab )*.")
        answer.Add("( b + ab )* represents
strings which do not contain any substring aa
and which end in b, and ( a + ab )* represents
strings which do not contain any substring bb.
Hence altogether it represents any string
consisting of a substring with no aa followed
by one b followed by a substring with no bb.")
        q.Add("Prove that DFA and NFA describe
the same regular language")
        answer.Add("Ask professor if answer is
correct.")
        q.Add("Prove union of two regular
language is regular")
        answer.Add("Ask professor is correct.")
    End Sub
End Class
```

Regex.vb

```
Public Class RegEx
    Public firstTerm As String = Nothing
    Public secondTerm As String = Nothing
    Public thirdTerm As String = Nothing
    Public fourthTerm As String = Nothing
End Class
```

RegularExpression.vb

```
Public Class Regular_Expression
    Private regex As String
    Private data As DataGridView
    Private transitions As ArrayList
    Private Origtransitions As ArrayList
    Private originputs As ArrayList
    Private inputs As ArrayList
    Private Start As String = ""
    Private Final As ArrayList
    Private RegularExpression As ArrayList
    Private frm As frm_REND

    Public Sub New(ByVal DFA As DataGridView,
ByVal f As frm_REND)
        frm = f
        data = DFA
        Final = New ArrayList
        RegularExpression = New ArrayList
        getStartFinal(data)
        transitions = New ArrayList
        Origtransitions = New ArrayList
        Origtransitions.Add("XX-" & Start)
        inputs = New ArrayList
        originputs = New ArrayList
        originputs.Add("")
        buildtransitions(data)

    End Sub

    Public Sub buildtransitions(ByVal data As
DataGridView)
        Dim trans As String = ""
        Dim input As String = ""
        For i As Integer = 0 To data.Rows.Count
- 2
```

```
            trans =
data.Rows(i).HeaderCell.Value & "-" &
data.Item(1, i).Value
            If data.Item(1, i).Value =
data.Item(2, i).Value Then
                input = "(" &
data.Columns(1).HeaderText & "+" &
data.Columns(2).HeaderText & ")"
                If Not trans.Contains("ø") Then
                    Origtransitions.Add(trans)
                    originputs.Add(input)
                End If

            Else
                input =
data.Columns(1).HeaderText
                If Not trans.Contains("ø") Then
                    Origtransitions.Add(trans)
                    originputs.Add(input)
                End If
                trans =
data.Rows(i).HeaderCell.Value & "-" &
data.Item(2, i).Value
                input =
data.Columns(2).HeaderText
                If Not trans.Contains("ø") Then
                    Origtransitions.Add(trans)
                    originputs.Add(input)
                End If

            End If
        Next
        getRegEx()
    End Sub

    Public Sub getRegEx()
        Dim temp As ArrayList = Origtransitions
        For i As Integer = 0 To Final.Count - 1
            transitions = temp.Clone()
            inputs = originputs.Clone()
            transitions.Add(Final(i).ToString &
"-" & "YY")
            inputs.Add("")
            frm.setText(vbCrLf & "Getting RE
From " & Start & " to " & Final(i))
            getTerm(transitions, Start,
Final(i).ToString)
        Next
        setConvertedResult()
    End Sub

    Public Sub setConvertedResult()
        Dim s As String
        frm.setTextRE("Clear")
        frm.setTextRE("The Regular Expression
equivalent of the given DFA is:")
        frm.setTextRE(vbCrLf &
RegularExpression(0))
        s = vbCrLf & RegularExpression(0)
        For i As Integer = 1 To
RegularExpression.Count - 1
            frm.setTextRE(vbCrLf & "+" &
RegularExpression(i))
            s = s & vbCrLf & "+" &
RegularExpression(i)
        Next
        Dim m As Message = New
Message("Success", "Conversion Successful!",
"Converting from DFA to Regular Expression",
"The Regular Expression equivalent of the given
DFA is:" & s, frm)
        m.Show()
    End Sub

    Public Sub getTerm(ByVal trans As
ArrayList, ByVal s As String, ByVal f As
String)
        Dim t As ArrayList = trans
        Dim z As Array
        Dim count As Integer = 0
        Dim total As Integer = t.Count
        While t.Count > 0
            t = arrange(t)
            For i As Integer = 0 To t.Count - 1
                Dim c As Array =
t(i).ToString.Split("-")
                If checkifLast(t, f) Then
                    getFinalRE(t, f)
                    Exit While
                ElseIf Not
t(i).ToString.Contains(f) Or (t(i).contains(f)
And Not t(i).contains("XX") And Not
t(i).contains("YY") And Not c(1).Equals(f))
Then
                    z =
t(i).ToString.Split("-")
                    Exit For
                End If
                count = count + 1
                If count > total * total +
total Then
                    Exit While
                End If
            Next

            Dim arr As ArrayList =
getConnectedStates(z(1), t)
            Dim a As New ArrayList
            Dim x = arr.Count - 1
            For i As Integer = 0 To x
                If Not
arr(i).ToString.Equals(z(1)) Then
                    a.Add(arr(i))
                End If
            Next

            t = EliminateState(t, z(0), z(1),
a, f)

        End While

    End Sub

    Public Function arrange(ByVal t As
ArrayList) As ArrayList
        Dim temp As New ArrayList
        Dim inp As New ArrayList
        For i As Integer = 0 To t.Count - 1
            Dim z As Array =
t(i).ToString.Split("-")
            If z(0).Equals("XX") Then
                temp.Add(t(i))
                inp.Add(inputs(i))
            End If
        Next

        For j As Integer = 0 To t.Count - 1
            If Not temp.Contains(t(j)) Then
                temp.Add(t(j))
                inp.Add(inputs(j))
            End If
        Next
        inputs = inp
        Return temp
    End Function

    Public Sub getFinalRE(ByVal t As ArrayList,
ByVal f As String)
        Dim stoe As String =
getTransitionInput("XX", f, t)
        Dim etoe As String =
getTransitionInput(f, f, t)
        Dim re As String = ""
        If Not stoe.Equals("") Then
            re = re & stoe
        End If
        If Not stoe.Equals("") Or Not
etoe.Equals("") Then
            If Not etoe.Equals("") Then
                re = re & etoe & "*"
            End If
        Else
```

```
            re = "Empty Language"
        End If

        RegularExpression.Add(re)
    End Sub

    Public Function checkifLast(ByVal t As
ArrayList, ByVal f As String) As Boolean
        Dim final As Boolean = True
        For i As Integer = 0 To t.Count - 1
            If t(i).Contains(f) Then
                If Not
t(i).ToString.Contains("XX") And Not
t(i).ToString.Contains("YY") And Not
t(i).Equals(f & "-" & f) Then
                    final = False
                End If
            End If
        Next
        Return final
    End Function
    Public Function EliminateState(ByVal t As
ArrayList, ByVal s As String, ByVal e As
String, ByVal a As ArrayList, ByVal f As
String) As ArrayList
        Dim tempinp As New ArrayList
        Dim tempTrans As New ArrayList
        frm.setText("Now Eliminating state..."
& e)
        For i As Integer = 0 To a.Count - 1
            Dim _final As String = a(i)
            Dim output As String = getInput(t,
s, e, _final, a, f)
        Next
        For j As Integer = 0 To t.Count - 1
            If Not t(j).ToString.Contains(e)
And Not tempTrans.Contains(t(j)) Then
                tempTrans.Add(t(j))
                tempinp.Add(inputs(j))
            End If
        Next
        t = tempTrans
        inputs = tempinp
        Return t
    End Function

    Public Function getInput(ByVal t As
ArrayList, ByVal s As String, ByVal e As
String, ByVal f As String, ByVal arr As
ArrayList, ByVal _final As String) As String
        Dim output As String = ""
        Dim stos As String =
getTransitionInput(s, s, t)
        Dim stof As String =
getTransitionInput(s, f, t)
        Dim ftos As String =
getTransitionInput(f, s, t)
        Dim stoe As String =
getTransitionInput(s, e, t)
        Dim etos As String =
getTransitionInput(e, s, t)
        Dim etoe As String =
getTransitionInput(e, e, t)
        Dim etof As String =
getTransitionInput(e, f, t)
        Dim ftoe As String =
getTransitionInput(f, e, t)
        Dim ftof As String =
getTransitionInput(f, f, t)
        If Not stof.Equals("") Then
            output = "(" & stof & "+"
        End If
        If Not stoe.Equals("") Then
            output = output & stoe
        End If

        If Not etoe.Equals("") Then
            output = output & etoe & "*"
        End If

        output = output & etof
        If Not stof.Equals("") Then
            output = output & ")"
        End If

        Dim found As Boolean = False
        'case1 ftoe not null s --- > e ----->
f-----> e
        If Not ftoe.Equals("") And Not
etof.Equals("") Then
            Dim str As String = ""
            'if "XX-f" exist then concatenate
            If f.Equals(_final) Then
                str = stoe
                If Not etoe.Equals("") Then
                    str = str & etoe & "*"
                End If
                str = str & etof
                For i As Integer = 0 To t.Count
- 1
                    If t(i).ToString.Equals(s &
"-" & f) Then
                        found = True
                        If Not
inputs(i).Equals(str) And Not
inputs(i).contains("+" & str) Then
                            inputs(i) = "(" &
inputs(i) & "+" & str & ")"
                            frm.setText("  " &
s & "->" & f & " := " & inputs(i))
                        Else
                        End If
                    End If
                Next
                If Not found Then
                    t.Insert(0, s & "-" & f)
                    inputs.Insert(0, str)
                    frm.setText("  " & s & "->"
& f & " := " & str)
                Else
                    found = False
                End If

                Dim looop As String = ""
                looop = ftoe
                If Not etoe.Equals("") Then
                    looop = looop & etoe & "*"
                End If
                looop = looop & etof

                For i As Integer = 0 To t.Count
- 1
                    If t(i).ToString.Equals(f &
"-" & f) Then
                        found = True
                        If Not
inputs(i).Equals(looop) And Not
inputs(i).ToString.Contains("+" & looop) Then
                            inputs(i) = "(" &
inputs(i) & "+" & looop & ")"
                            frm.setText("  " &
f & "->" & f & ":=" & inputs(i))
                        End If
                    End If
                Next
                If Not found Then
                    t.Insert(0, f & "-" & f)
                    inputs.Insert(0, "(" &
looop & ")")
                    frm.setText("  " & f & "->"
& f & ":=" & "(" & looop & ")")
                End If
            Else
                str = ""
                For i As Integer = 0 To t.Count
- 1
                    If t(i).ToString.Equals(s &
"-" & f) Then
                        found = True
                        str = stoe
                        If Not etoe.Equals("")
Then
```

```
                            str = str & etoe &
"*"
                        End If
                        str = str & str & "("
                        str = str & ftoe & etof
& ")*"
                        If Not
inputs(i).Equals(str) And Not
inputs(i).contains("+" & str) Then
                            inputs(i) = "(" &
inputs(i) & "+" & str & ")"
                            frm.setText("  " &
s & "->" & f & ":=" & inputs(i))
                        End If
                    End If
                Next
                If Not found Then
                    Dim st As String = stoe
                    If Not etoe.Equals("") Then
                        st = st & etoe & "*"
                    End If
                    t.Insert(0, s & "-" & f)
                    st = st & etof & "(" & ftoe
                    If Not etoe.Equals("") Then
                        st = st & etoe & "*"
                    End If
                    st = st & etof & ")*"
                    inputs.Insert(0, st)
                    frm.setText("  " & s & "->"
& f & ":=" & st)
                Else
                    found = False
                End If

                Dim arrayofConnectedToState As
ArrayList = getCOnnectedTo(e, f, t)
                Dim a As ArrayList =
arrayofConnectedToState
                If Not a Is Nothing And a.Count
> 0 Then
                    For j As Integer = 0 To
a.Count - 1
                        Dim ntoe As String =
getTransitionInput(a(j), e, t)
                        If Not etoe.Equals("")
Then
                            ntoe = ntoe & etoe
& "*"
                        End If
                        ntoe = ntoe & etof &
"(" & ftoe
                        ntoe = ntoe & etof &
")*"

                        found = False

                        For i As Integer = 0 To
t.Count - 1
                            If t(i).Equals(a(j)
& "-" & f) Then
                                found = True
                                inputs(i) =
ntoe
                                frm.setText("
" & a(j) & "->" & f & ":=" & inputs(i))
                            End If
                        Next
                        If Not found Then
                            t.Insert(0, a(j) &
"-" & f)
                            inputs.Insert(0,
ntoe)
                            frm.setText("  " &
a(j) & "->" & f & ":=" & ntoe)
                        Else
                            found = False
                        End If
                    Next
                End If
```

```
                Dim arrayofConnectedStates As
ArrayList = getConnectedStates(e, f, t)
                a = arrayofConnectedStates
                If Not a Is Nothing And a.Count
> 0 Then
                    For j As Integer = 0 To
a.Count - 1
                        Dim eton As String = ""
                        If Not etoe.Equals("")
Then
                            eton = eton & etoe
& "*"
                        End If
                        eton = eton & "(" &
etof
                        If Not ftof.Equals("")
Then
                            eton = eton & ftof
& "*"
                        End If
                        eton = eton & ftoe &
")*"
                        eton = eton &
getTransitionInput(e, a(j), t)

                        found = False

                        For i As Integer = 0 To
t.Count - 1
                            If t(i).Equals(e &
"-" & a(j)) Then
                                found = True
                                inputs(i) =
eton
                                frm.setText("
" & e & "->" & a(j) & ":=" & eton)
                            End If
                        Next
                        If Not found Then
                            t.Insert(0, e & "-"
& a(j))
                            inputs.Insert(0,
eton)
                            frm.setText("  " &
e & "->" & a(j) & ":=" & eton)
                        Else
                            found = False
                        End If
                    Next

                End If

            End If

        Else 'case 2 s---> e ----> f
            For i As Integer = 0 To t.Count - 1
                If t(i).ToString.Equals(s & "-"
& f) Then
                    found = True
                    inputs(i) = output
                    frm.setText("  " & s & "->"
& f & ":=" & output)
                End If
            Next
            If Not found Then
                t.Insert(0, s & "-" & f)
                inputs.Insert(0, output)
                frm.setText("  " & s & "->" & f
& ":=" & output)
            End If
            Dim arrayofConnectedToState As
ArrayList = getCOnnectedTo(e, arr, t)
            Dim a As ArrayList =
arrayofConnectedToState
            If Not a Is Nothing And a.Count > 0
Then
                For j As Integer = 0 To a.Count
- 1
                    Dim ntoe As String =
getTransitionInput(a(j), e, t)
                    If Not etoe.Equals("") Then
```

```
                        ntoe = ntoe & etoe &
"*"
                    End If
                    ntoe = ntoe & etof

                    found = False

                    For i As Integer = 0 To
t.Count - 1
                        If t(i).Equals(a(j) &
"-" & f) Then
                            found = True
                            inputs(i) = ntoe
                            frm.setText("  " &
a(j) & "->" & f & ":=" & ntoe)
                        End If
                    Next
                    If Not found Then
                        t.Insert(0, a(j) & "-"
& f)
                        inputs.Insert(0, ntoe)
                        frm.setText("  " & a(j)
& "->" & f & ":=" & ntoe)
                    Else
                        found = False
                    End If
                Next
            End If
        End If



        Return output
    End Function

    Public Function notCOnnected(ByVal e As
String, ByVal f As String, ByVal t As
ArrayList)
        For i As Integer = 0 To t.Count - 1
            Dim z As Array =
t(i).ToString.Split("-")
            If z(0).Equals(f) And Not
z(1).Equals(e) Then
                Return False
            End If
        Next
        Return True
    End Function
    Public Function getCOnnectedTo(ByVal e As
String, ByVal f As String, ByVal t As
ArrayList) As ArrayList
        Dim a As New ArrayList
        For i As Integer = 0 To t.Count - 1
            Dim z As Array =
t(i).ToString.Split("-")
            If z(1).Equals(e) And Not
z(0).Equals(f) And Not z(0).Equals("XX") And
Not z(0).Equals(z(1)) Then
                a.Add(z(0))
            End If
        Next
        Return a
    End Function

    Public Function getTransitionInput(ByVal s1
As String, ByVal f1 As String, ByVal t As
ArrayList) As String
        For i As Integer = 0 To t.Count - 1
            Dim z As Array =
t(i).ToString.Split("-")
            If z(0).ToString.Equals(s1) And
z(1).ToString.Equals(f1) Then
                Return inputs(i)
                Exit Function
            End If
        Next
        Return ""
    End Function
    Public Function getConnectedStates(ByVal
elimState As String, ByVal f As String, ByVal t
As ArrayList) As ArrayList
        Dim arr As New ArrayList
        Dim x As Boolean = True
        For i As Integer = 0 To t.Count - 1
            Dim z As Array =
t(i).ToString.Split("-")
            x = True
            If z(0).Equals(elimState) And Not
z(1).Equals(f) And Not z(1).Equals("YY") And
Not z(1).Equals(z(0)) Then
                arr.Add(z(1))
            End If
        Next
        Return arr
    End Function

    Public Function getConnectedto(ByVal
elimState As String, ByVal a As ArrayList,
ByVal t As ArrayList) As ArrayList
        Dim arr As New ArrayList
        Dim x As Boolean = True
        For i As Integer = 0 To t.Count - 1
            Dim z As Array =
t(i).ToString.Split("-")
            x = True
            If z(1).Equals(elimState) Then
                For j As Integer = 0 To a.Count
- 1
                    If z(0).Equals(a(j)) Or
z(0).Equals("XX") Or z(0).Equals(z(1)) Then
                        x = False
                    End If
                Next
                If x Then
                    arr.Add(z(0))
                End If

            End If
        Next

        Return arr
    End Function

    Public Function getConnectedStates(ByVal
elimState As String, ByVal t As ArrayList) As
ArrayList
        Dim a As New ArrayList
        For i As Integer = 0 To t.Count - 1
            Dim z As Array =
t(i).ToString.Split("-")
            If z(0).Equals(elimState) And Not
z(1).Equals("YY") And Not
z(1).Equals(elimState) Then
                a.Add(z(1))
            End If
        Next
        Return a
    End Function

    Public Sub getStartFinal(ByVal d As
DataGridView)

        Start = getStartState()
        Final = getFinalState()
    End Sub

    Public Function getStartState() As String
        For i As Integer = 0 To data.Rows.Count
- 2
            If data.Item(0,
i).Value.ToString.Contains("Start") Then
                Return
data.Rows(i).HeaderCell.Value
            End If
        Next
        Return Nothing
    End Function

    Public Function getFinalState() As
ArrayList
        Dim a As New ArrayList
        For i As Integer = 0 To data.Rows.Count
- 2
            If data.Item(0,
i).Value.ToString.Contains("Final") Then
                a.Add(data.Rows(i).HeaderCell.V
alue)
            End If
        Next
        If a.Count > 0 Then
            Return a
        End If
        Return Nothing
    End Function
End Class
```

State.vb

```
Imports Microsoft.DirectX
Imports Microsoft.DirectX.Direct3D
Imports Direct3D = Microsoft.DirectX.Direct3D
Public Class State
    Public inputs As New ArrayList
    Public toState As New ArrayList
    Public Name As String = Nothing
    Public Ref As String = Nothing

    Private vertices As
CustomVertex.PositionColored()

    Public radius As Double = 1.5
    Public x_Start1 As Double = 0
    Public y_Start1 As Double = 0
    Public z_Start1 As Double = 0
    Public x_Start2 As Double = 0
    Public y_Start2 As Double = 0
    Public z_Start2 As Double = 0
    Public force1 As Double = 0
    Public force2 As Double = 0
    Public color As Color = color.Green
    Public kind As String = Nothing


    Public Sub New()
        vertices = New
CustomVertex.PositionColored(1) {}
    End Sub


    Public Sub setVertices(ByVal vertex1 As
Double, ByVal vertex2 As Double, ByVal vertex3
As Double, ByVal text As String)
        If text.Equals("(0)") Then
            x_Start1 = vertex1
            y_Start1 = vertex2
            z_Start1 = vertex3
        End If
        If text.Equals("(1)") Then
            x_Start2 = vertex1
            y_Start2 = vertex2
            z_Start2 = vertex3
        End If

    End Sub


    Public Sub setAttribute()
        vertices(0).SetPosition(New
Vector3(x_Start1, y_Start1, z_Start1))
        vertices(0).Color =
Color.DarkSeaGreen.ToArgb
        vertices(1).SetPosition(New
Vector3(x_Start2, y_Start2, z_Start2))
        vertices(1).Color = Color.Green.ToArgb
    End Sub


    Public ReadOnly Property getVertices() As
CustomVertex.PositionColored()
        Get
            Return vertices
        End Get
    End Property


End Class
```

Text.vb

```
Public Class Text
    Public x_Pos As Double = 0
    Public y_Pos As Double = 0
    Public z_Pos As Double = 0
    Public z_Yaw As Double = 0
    Public _text As String = Nothing

    Public Sub setAttribute(ByVal x As Double,
ByVal y As Double, ByVal z As Double, ByVal yaw
As Double, ByVal text As String)
        x_Pos = x : y_Pos = y : z_Pos = z :
z_Yaw = yaw * Math.PI / 180 : _text = text
    End Sub
End Class
```

XI. Acknowledgement

First and foremost I would like to thank our father ,my bestfriend, in Heaven who inspired me a lot and I know who guided me throughout the thesis proper and my study. I would like to extend my deepest gratitude to my family who serve as my inspiration in my studies, to my parents, Marites Pasco and Fermino Pasco, who gave me moral support, love and care, and of course financial support, to my handsome brother and pretty sisters, Xian Paul, Ytle and Fei, who gave me encouragement in when im having a hard time doing my responsibilities. To my Lola Agring, thankyou for your advices, care and love.

I would like to thank and extend my appreciation to the beautiful Miss Jolly Pearl Gomez, my best friend, my close friend, my teacher, ultimate listener, my everything, who inspires me a lot during my studies. Thankyou for being there at all times. Thankyou for understanding me when I'm in trouble. Also, thankyou for the printer which I used in prepairing my theisis paper. Thankyou for all the memories, it may be good or bad, but it is still worth taking if its with you.

This work may not be possible without the help of my adviser , Dr. Vincent Peter Magboo, thank you for your guidance and support while I'm doing my thesis. Thankyou for being patient and considerate. Thankyou for suggesting some enhancements in my system.

To all my blockmates and my acquaintances at school, thankyou for being part of my life. To my closest friends, Donnel (thankyou for a low price book binding and all your help and encouragement), Mike(thankyou for moral support and alert to all the things I need to do), April(since elementary, highschool, and today you remain a good family friend), Rac(thankyou for support and sharing your thoughts), Roy(for snappy approach yet useful), Jenzen(for letting me play your PSP and support), Mark (thankyou for sharing your thesis paper for my reference), Dave(though your in a far away land, you still find a way to reach us). To all my friends thankyou! Ate Cha, my dear cousin, thankyou for guidance and materials that I borrowed.

To my housemates Essel, Christian, Evan, thankyou for the company. Essel, thankyou for lending me your laptop when I needed it most. Evan, thankyou for letting me store some files in your External Drive. Christian, thankyou for the moral support and company while I'm doing my thesis. Thankyou animales for the support and memories.

Really can't leave the house because of the good times(playing DOTA, singing, wrestling, basketball, watching movies, playing guitar, EATING)

To my highschool friends who are near yet so far. To Esper, Owen, Wau, Carlo, Patrick, Dope, Kim, Maan, Mei, RoseAnne, Mia, Tristan, Tantan, Libby, Leonard, Nilo, Paula, Shyr, Ive, Jai, Kenneth, Kevin, Eugene, JC, Yen, Olive, Alger, Gemmae, Jyko, Neil, Paolo, etc., thankyou for the memories and thankyou for staying in my life.

To my elementary colleagues, Bobe, Jeff, Euland, Yuri, thankyou for the support and encouragement.

I cannot end this without extending my gratitude to my professors from first year to fourth year. Thankyou for the lessons I learned. And also to Ate Eden, thankyou for entertaining all my questions when I'm lost.

Thankyou to all the people who cares and whom I care☺