TIC 2151 THEORY OF COMPUTATION
TRIMESTER 2 2018/2019

GROUP ASSIGNMENT

Prepared By

Tutorial Section: TT01 / Group 1

Group Members	Student ID	Contribution Percentage
Wayile Jialade ***(Leader)***	1151102347	25%
Samuel Wong Ing Shiing	1151103687	25%
Teoh Chin Wei	1161304075	25%
Chew Shi Liang	1161304221	25%

Prepared To

Dr. Nbhan D. Salih

Due Date of Submission: 11 Feb 2019 (Week 13)

TABLE OF CONTENT

1.0 Introduction

The objective of this assignment is to test the understanding of student on how to convert the concepts of the subjects 'Theory of Computation'(TIC2151) which is lead by Dr.Nbhan.D.Salih into a computer program. In this case, the knowledge of **visual basic programming** will be applied to presenting the **Graphical User Interface (GUI) program** to the users.

In this report, there are two parts of the program will be discussed which are stated below. In this case, the design flow chart and manual to operate the program will be explained as detail.

- **Part 1:** Conversion between Regular Grammar & Non-Deterministic Finite Automaton.

- **Part 2:** Convert the Context-Free Grammar(CFG) into the Chomsky Normal Form(CNF) & produce a CYK chart based on previous CNF.

A **regular grammar** is a formal grammar that is either left-regular or right-regular and every regular grammar describes a regular language. Regular grammar is a more powerful way to describe languages than finite automata. An NFA or Non-Deterministic Finite Automaton is a finite state machine which does not need to have each of its transitions uniquely determined by its source state and input symbol, as well as not needing the reading of an input symbol for each state transition.

A **context-free grammar** is an even more powerful method to describe languages in a way that it can describe certain features that have a recursive structure which is crucial in a certain application. Generally, context-free grammar often can have multiple simplified forms and one of the simplest and most useful forms is called the Chomsky normal form. A context-free grammar is successfully converted into the Chomsky normal form when every rule of the form is in; A -> BC, A -> a where a is any terminal and A, B, and C are any variables except that B and C will not be the start variable. To determine if a word generated is part of a grammar, given a CNF grammar.

2.0 Design Flowcharts

PART I :

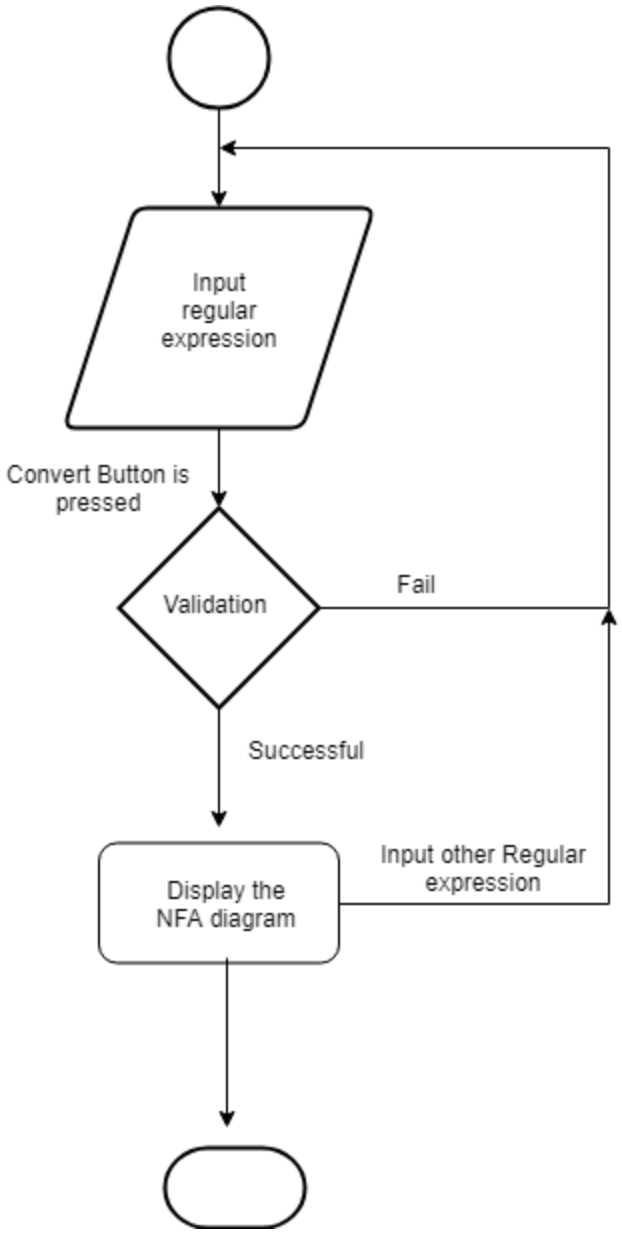


Figure 1: Flow Chart of Part 1 Feature

PART II :

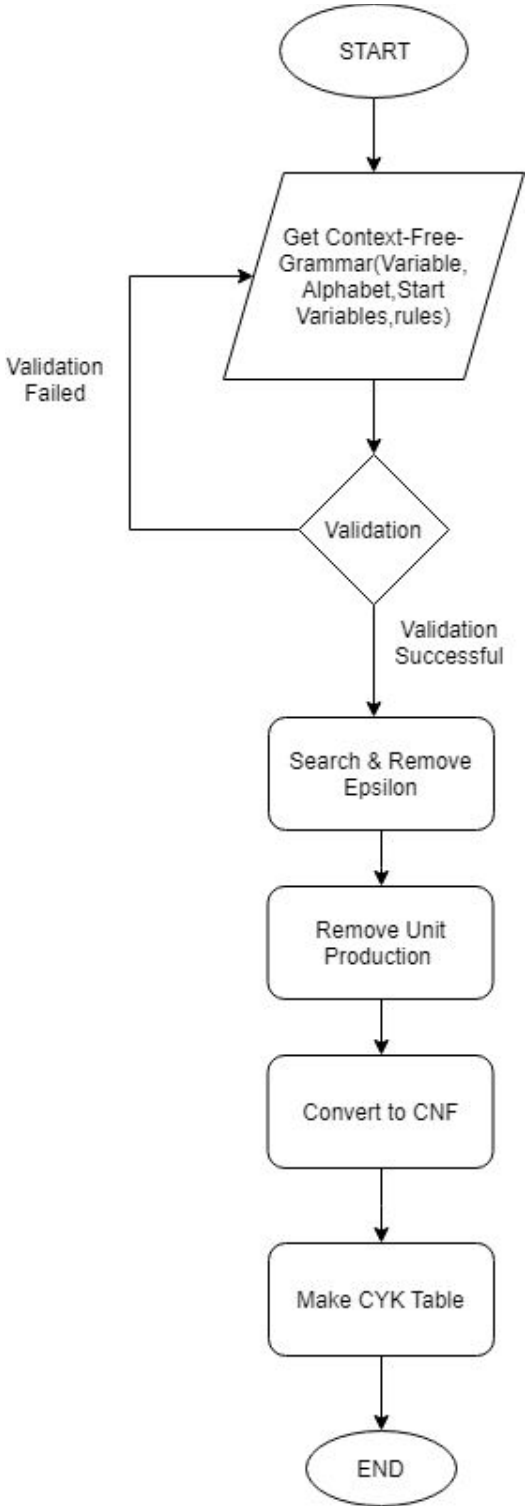
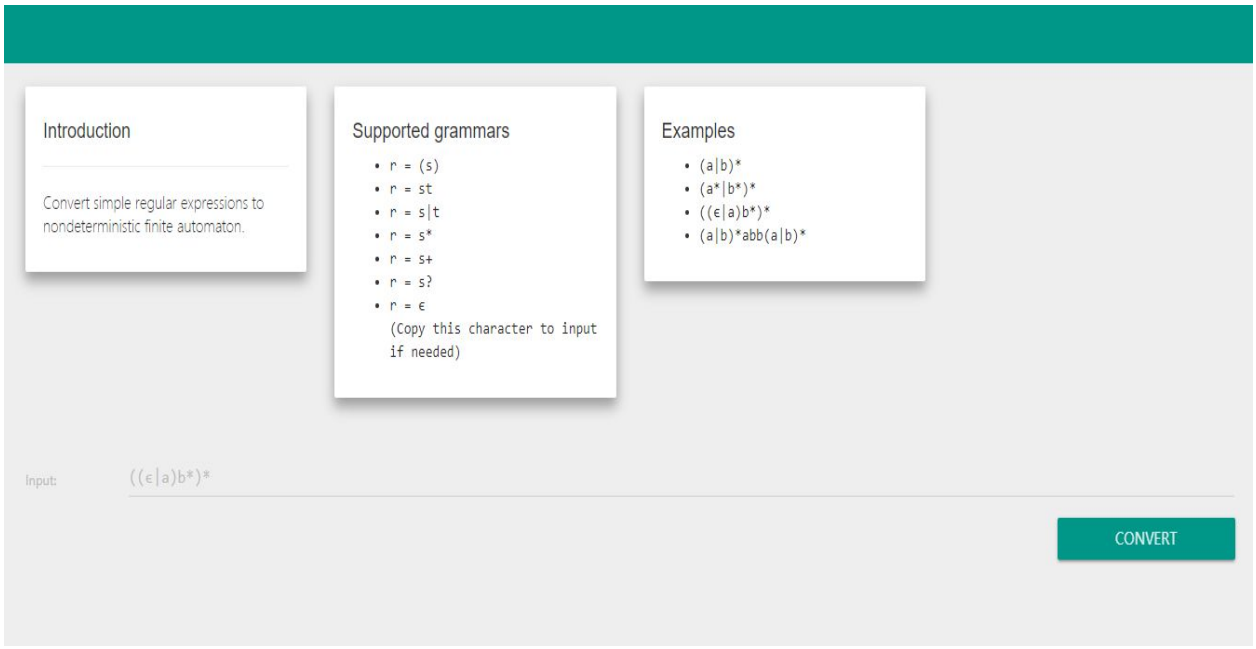


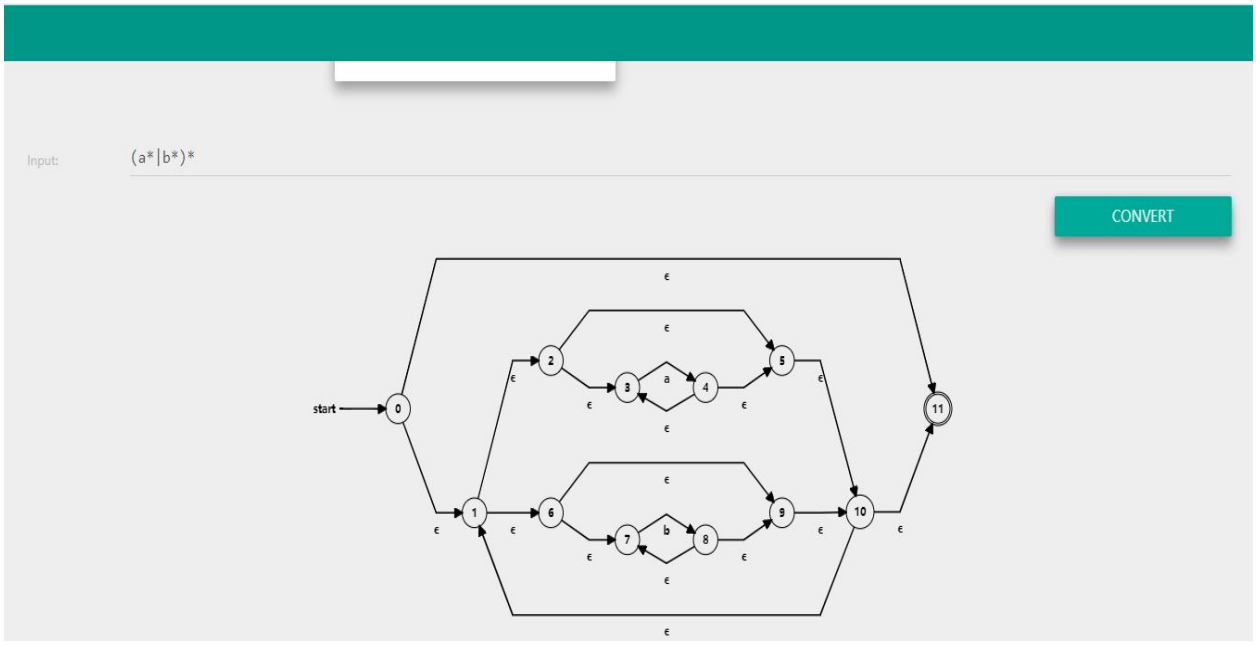
Figure 2: Flow Chart of Part 2 Feature

3.0 Screenshots

PART I:

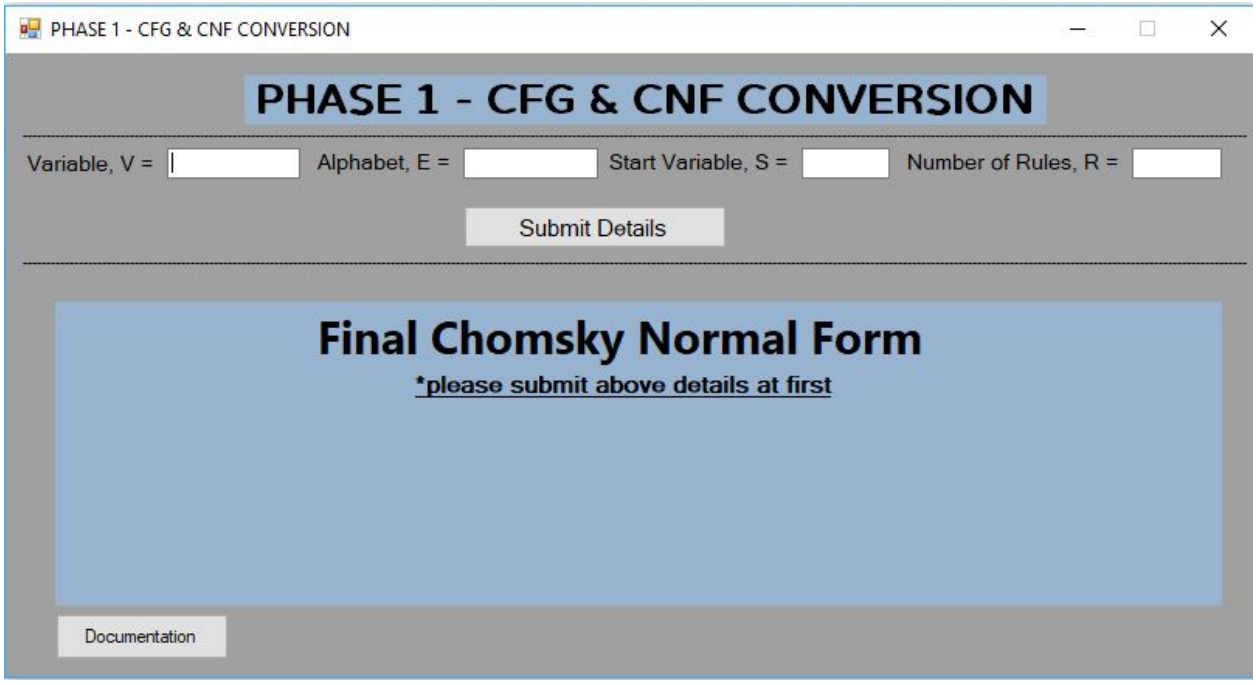


(Screenshot 1: Main Page of Part 1 Feature)

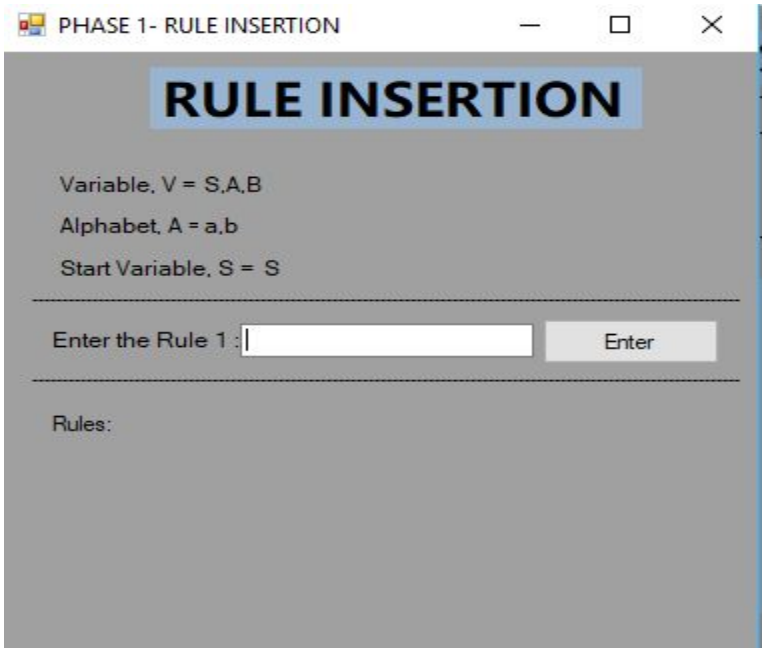


(Screenshot 2: Result Displayed of Part 1 Feature)

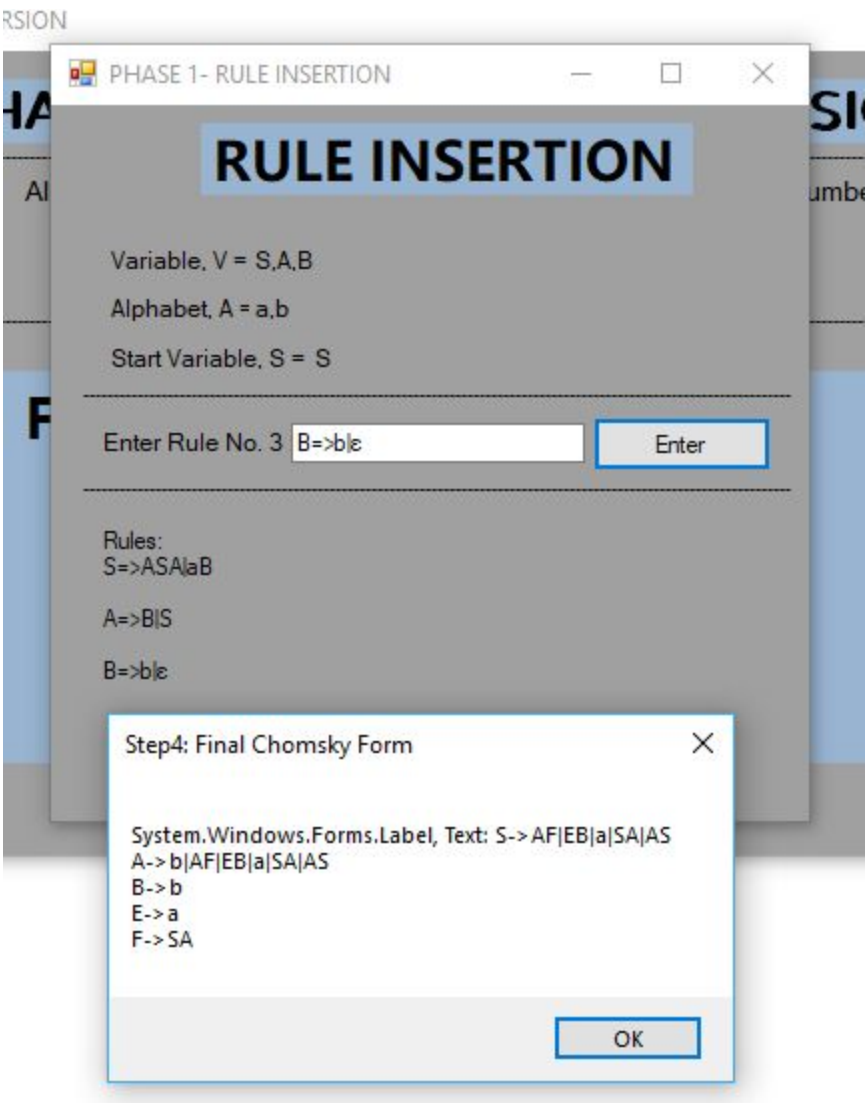
PART II:



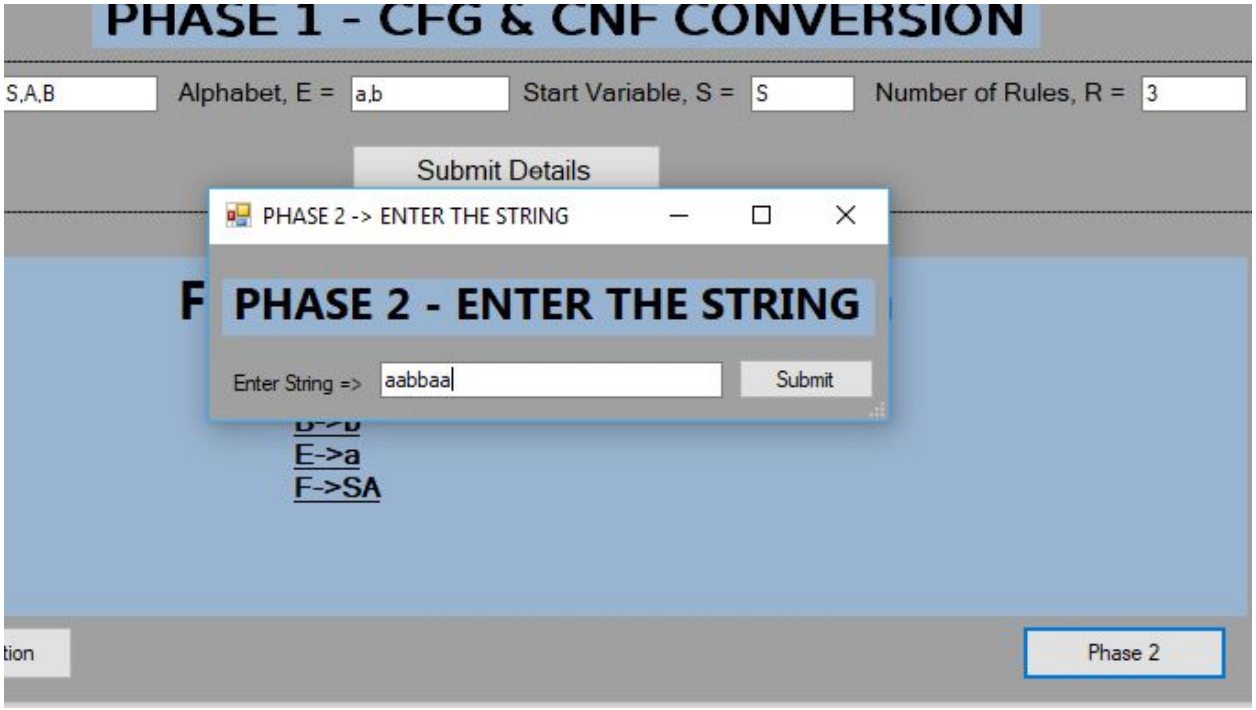
(Screenshot 1: The CFG Input Layout)



(Screenshot 2: Rule Insertion Layout)



(Screenshot 3: CFG - CNF Conversion)



(Screenshot 4: Phase2 Enter the string)

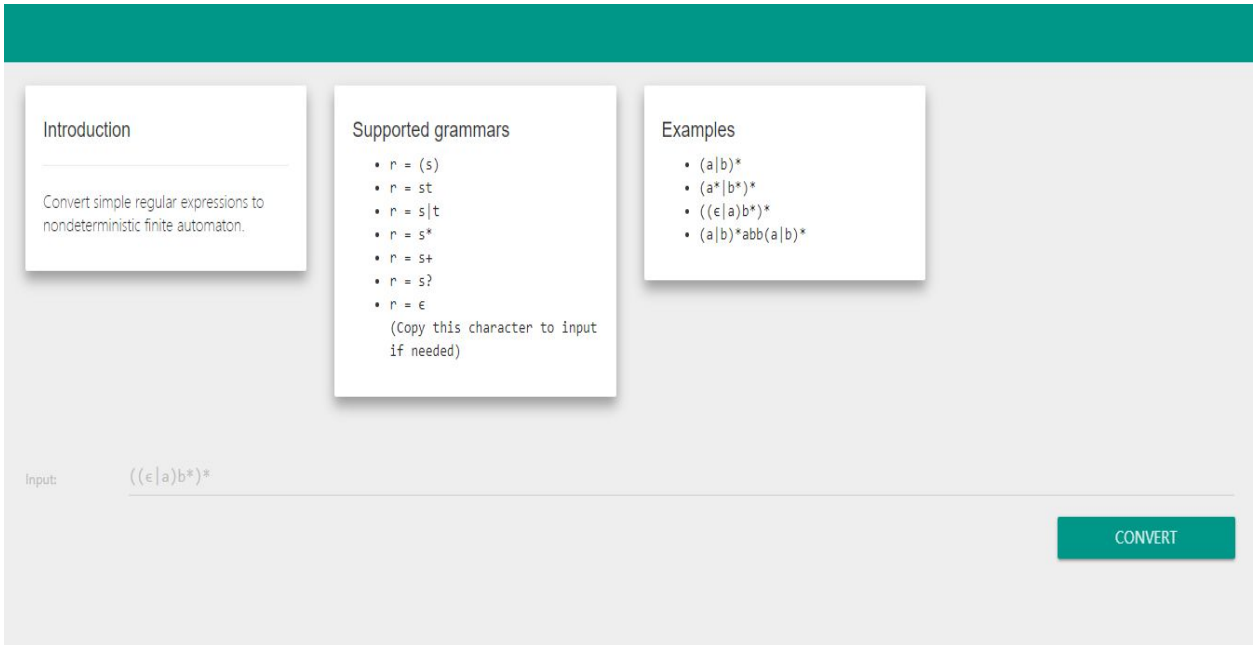
4.0 Manual with Examples

Figure 3: Main Menu

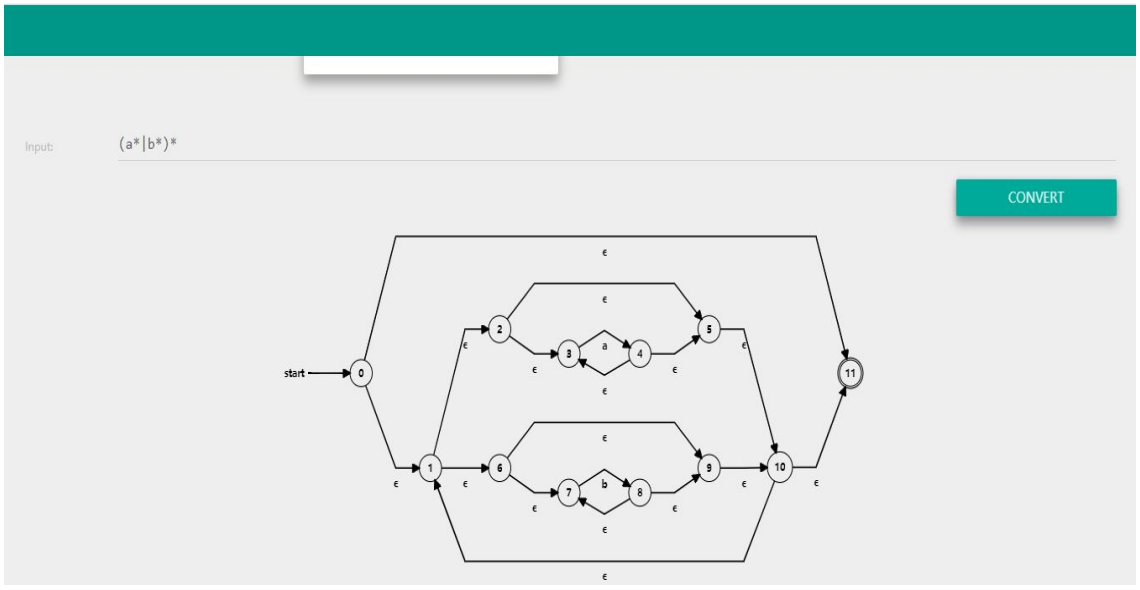
- Click to open the program '**TIC2151-Assignment.exe'**, and you will see the interface like the image above.
- In the main menu, the user is required to choose and click respective buttons for entering the feature of part 1, part 2 & also the documentation.

PART I:

At first, the user will direct into the page that illustrates as below after button ‘Part 1’ is clicked. The user needs to input the regular expression into an input text box and then press the button to conduct conversion into NFA diagram.



After the button ‘Convert ‘ is pressed, the regular expression entered will undergo validation on checking the regular expression entered whether correct. If the validation is correct, the NFA diagram will be displayed.



If validation is failed, an error message will be displayed to alert the user to enter again.

PART II:

Step 1: First of all, the user will see the main interface like the image below

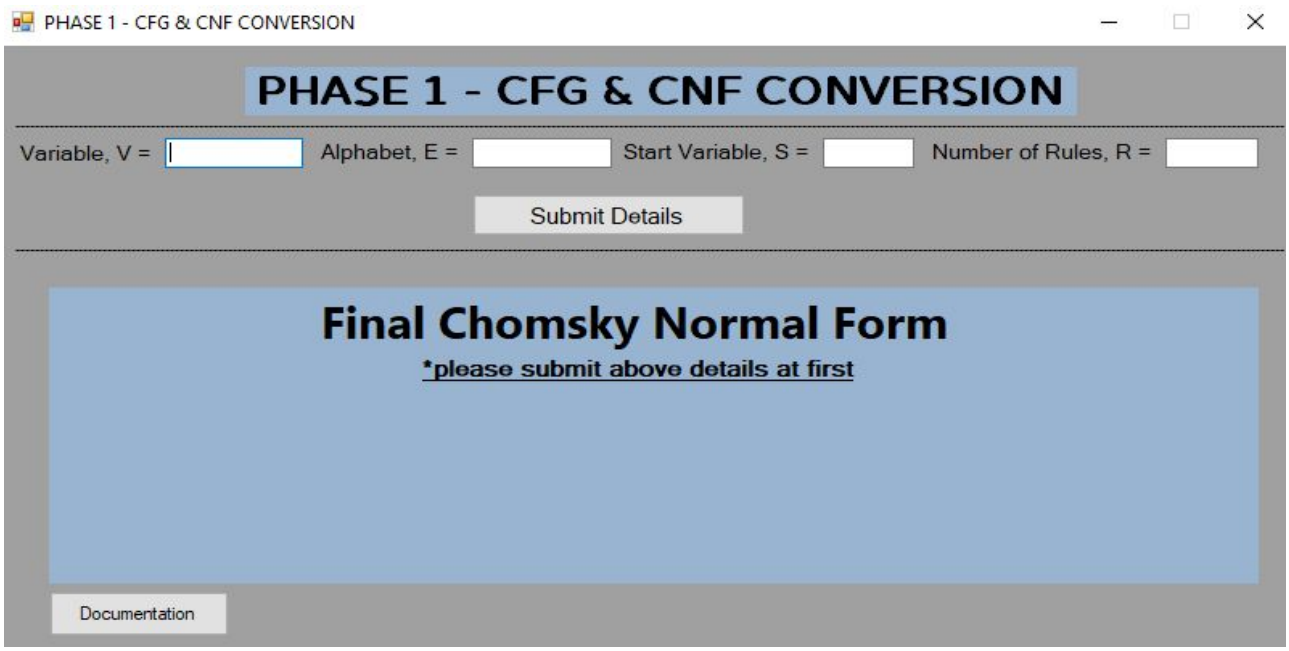


Figure 3: Interface of Phase 1- CFG->CNF conversion

Step 2: In order to input the Context-Free_Grammar correctly, user need to input of variable, alphabet, start variable and number of rules, therefore you need to type all of them into respective textbox, if user need to type more than one letter inside the textbox, make sure separate each of the letters with comma, image below can be as correct guideline, e.g, if you want to input a CF grammar

S => ASA | aB

A => B | S

B => b | e

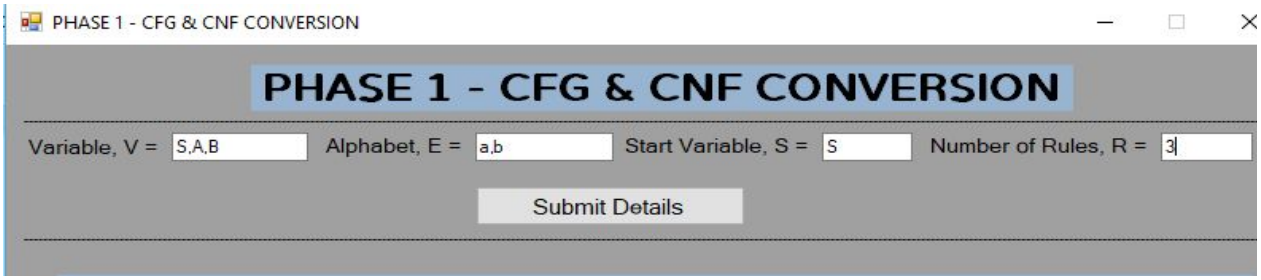


Figure 4: Procedure to Input The Data Accurately

Step 3:- Now you can click the button 'Submit Details', you will see a new window form pop out, this form is used type down the rules, the interface will look like this:

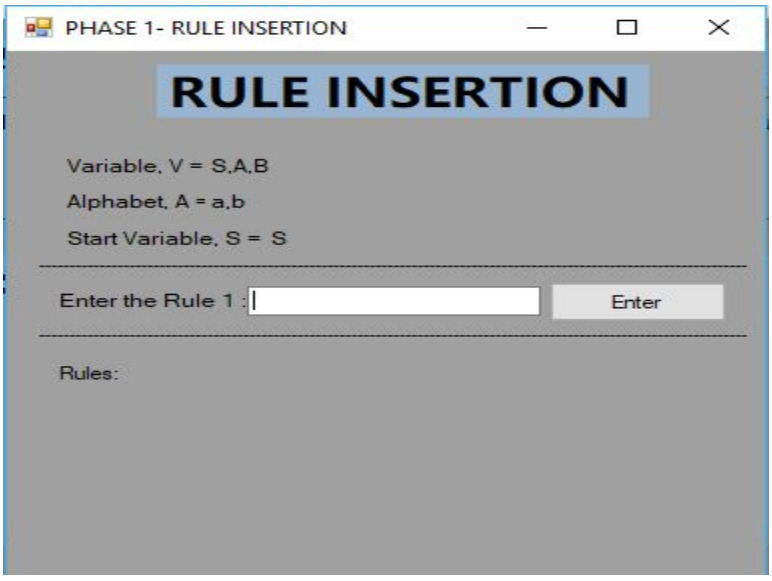


Figure 5: The interface for Rule Insertion

Step 4: It's time to enter the rule, for the sake of illustration, we will still use the example grammar after all the rules have entered, the program will carry out the transform automatically, each step of the conversion will be displayed at a pop-out dialogue as the following images shown.

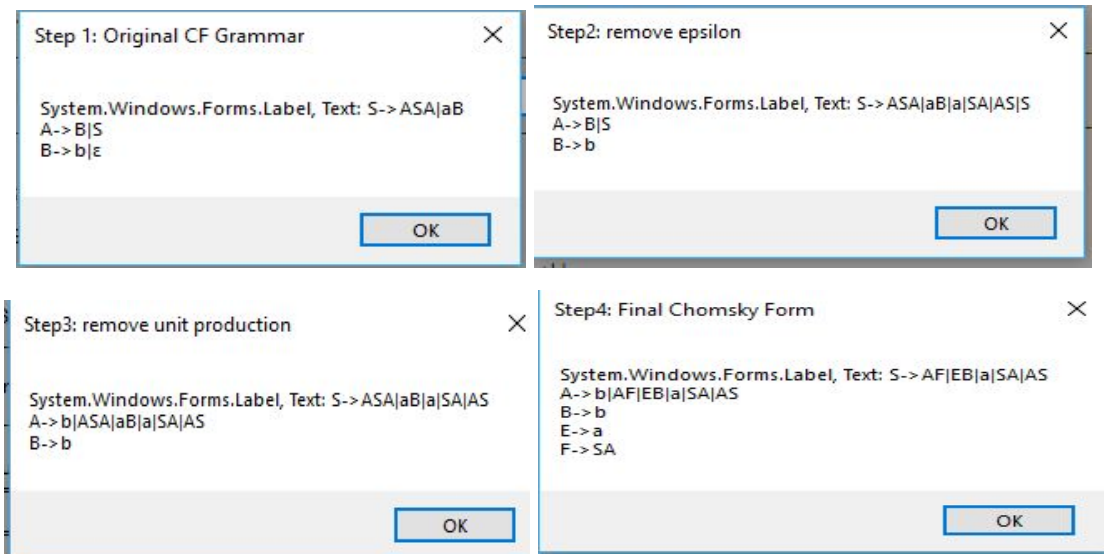


Figure 6: Four steps of conversion

Step 5: Now you got the final CNF of the grammar as the following images shown.

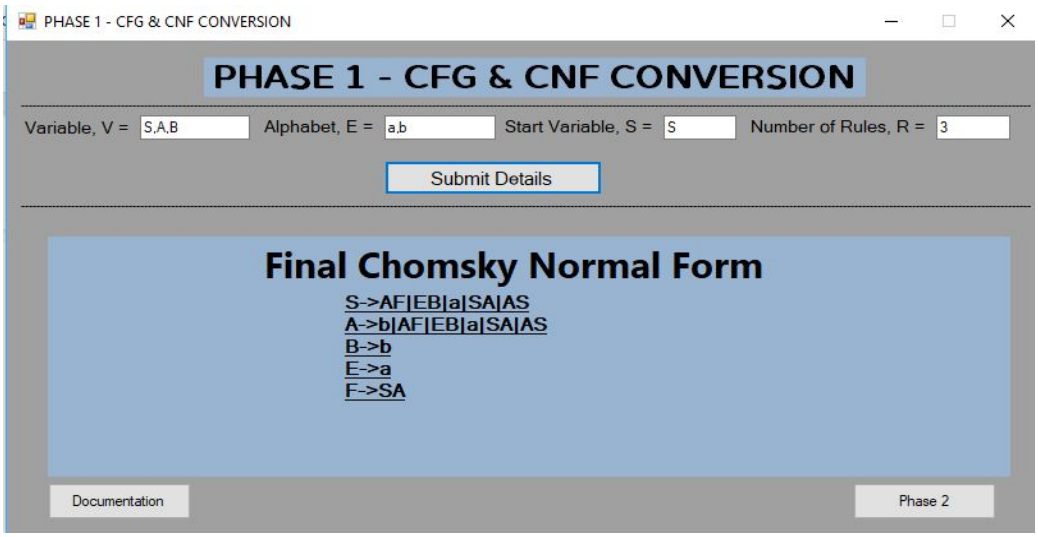


Figure 7: Output Displayed for final CNF

Step 6: Here you can click for Phase 2 which is used to make a CYK table based on the CNF you get when you enter the string, you can only type in the letter which you typed in to the textbox of Alphabet, e.g:

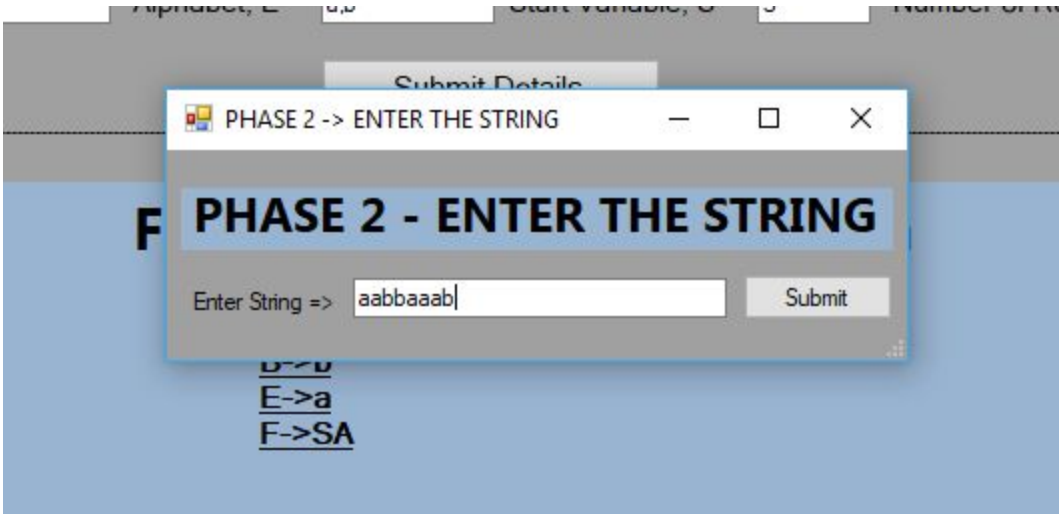


Figure 8: The string can be any combination of the letters, while the letters have to be the one which you entered in the textbox of 'Alphabet'

Step 7: Now you can get the CYK table by simply submit the string you entered.

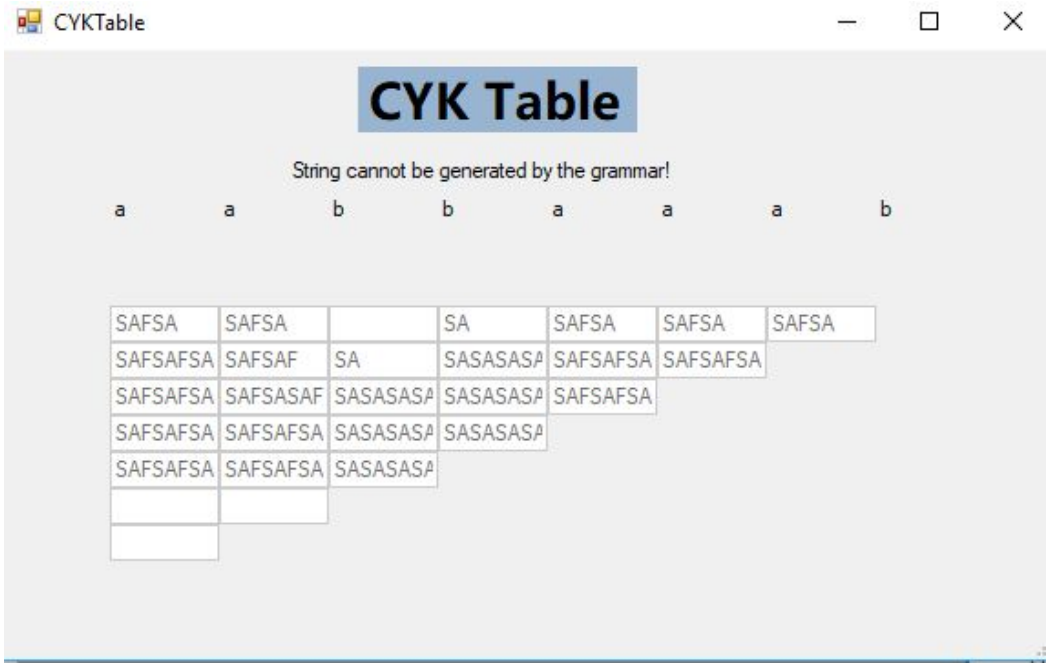


Figure 9: The CYK Table based on your CNF & the string.

5.0 Important Codes

search_epsilon function:

```
Sub searchEpsilon()
        current_node = head_node
        current_row = head_node
        Dim a As Integer = 0
        Dim last_node As Node

        While Not current_row Is Nothing
            While Not current_node.child Is Nothing
                last_node = current_node
                current_node = current_node.child
                If current_node.info = "ε" Then
                    variable_with_epsilon(a) = current_row.info
                    a += 1

                    If Not current_node.child Is Nothing Then
                        Dim temp As Node
                        temp = current_node.child
                        last_node.child = temp
                        current_node = last_node.child
                    Else
                        current_node = Nothing
                        last_node.child = Nothing
                    End If
                    Exit While
                End If
            End While
            current_row = current_row.pnext
            current_node = current_row
  End While
End Sub
```

remove_epsilon function:

```
Sub removeEpsilon()
        current_node = head_node
        current_row = head_node
        Dim lastNode As Node
        epsilon_exist = False
        Dim checkNode As Node

 For i = 0 To variable_with_epsilon.Length - 1
            While Not current_row Is Nothing
                node_exist = False
                While Not current_node.child Is Nothing
                    current_node = current_node.child
                    lastNode = current_node
                    Dim eachChar() As Char = current_node.info.ToCharArray()

 For j = 0 To eachChar.Length - 1
                        For m = 0 To variable_with_epsilon.Length - 1
                            If eachChar(j) = variable_with_epsilon(m) Then
                                Dim newV As String = ""
                                checkNode = current_row

                                If eachChar.Length > 1 Then
                                    For k = 0 To eachChar.Length - 1
                                        If j <> k Then
                                            newV = newV + eachChar(k)
                                        End If
                                    Next

                                Else
                                    newV = "ε"
                                    'If current_row.info <> start_variable Then
                                    epsilon_exist = True
                                End If

                                While Not checkNode.child Is Nothing
                                    checkNode = checkNode.child
                                    If checkNode.info = newV Then
                                        node_exist = True
                                    End If
                                End While

                                If node_exist = False Then
                                    While Not lastNode.child Is Nothing
                                        lastNode = lastNode.child
                                    End While
```

```
                                    lastNode.child = New Node
                                    lastNode = lastNode.child
                                    lastNode.info = newV
                                End If
                            End If
                        Next

                    Next
                End While
current_row = current_row.pnext
current_node = current_row
            End While
        Next
    End Sub
```

rule_insertion function:

```
‘Make LinkedList for each string stored’
PARTII.current_node = PARTII.head_node
        If PARTII.head_node.info = first_letter Then    'to check if the Variable is
the start node
            For j = 0 To temp.Length - 1
                PARTII.current_node.child = New Node
                PARTII.current_node = PARTII.current_node.child
                PARTII.current_node.info = temp(j)
                'MsgBox(PARTII.current_node.info.ToString)
            Next
        Else
            While Not PARTII.current_node.pnext Is Nothing
                PARTII.current_node = PARTII.current_node.pnext
            End While

            PARTII.current_node.pnext = New Node
            PARTII.current_node = PARTII.current_node.pnext
            PARTII.current_node.info = first_letter

            For j = 0 To temp.Length - 1
                PARTII.current_node.child = New Node
                PARTII.current_node = PARTII.current_node.child
                PARTII.current_node.info = temp(j)
            Next
        End If
```

==========

Thank You