COMP 4400 Automata Theory
Project#2 – Implementing an NPDA
Due Date: 04/28/24

Overview:

You will need to implement a general purpose NPDA in C++ that can read the description of the NPDA from an input file and construct the NPDA. The input file will be assumed to store the NPDA description in a standard format. The content of a sample file describing an NPDA that accepts the language $\{a^n b^n : n \geq 0\}$ is shown next:

```
<states>
q0
q1
q2
q3
<input alphabet>
a
b
<stack alphabet>
0
1
<transitions>
q0 a 0 q1 10
q0 * 0 q3 *
q1 a 1 q1 11
q1 b 1 q2 *
q2 b 1 q2 *
q2 * 0 q3 *
<initial state>
q0
<stack start>
0
<final states>
q3
```

You will modify and extend the DFA class from Project#1 to create the NPDA class. The NPDA class will have the following data:

- a vector of states where each state will be represented as a string
- a vector of alphabet symbols where each symbol will be represented as a character
- a vector of stack alphabet symbols where each symbol will be represented as a character
- a map (key-value) of transitions $\left(\delta: Q\times(\Sigma \cup \{\lambda\})\times\Gamma \rightarrow (Q\times\Gamma^*)\right)$ where the "key" will be a 3-tuple (a string representing the current state of the automata, a character representing the input symbol being processed, and a character representing the stack top symbol) and the "value" will be a vector of pairs (each pair is a string representing the next state and a string representing the sequence of stack symbols that will replace the stack top)
- a string representing the initial state
- a character representing the stack start symbol
- a vector of final states where each final state will be represented as a string
- a string representing the current state of the automata

The NPDA class will have the following member functions

- a function for adding a given state to the automata
- a function for adding a given alphabet symbol
- a function for adding a given stack alphabet symbol
- a function to add a transition (<state,symbol,stack top> to <state,replace string>)
- a function to set the initial state
- a function to set the stack start symbol
- a function to add a final state
- a function to display the NPDA description (as shown in the sample execution)
- a function to set the current state to the initial state
- a function to make a transition from the current state to the next state for a given input symbol and stack top symbol
- a function that will return the current state of the automata
- a function to check if a symbol is valid
- a function to check if a given state is a final state

The driver code should read a given input file (whose name will be provided from command line) and instantiate the NPDA. Then the program should display the NPDA description in the format shown in the sample execution. Then the user will be continually prompted to enter an input string (until Ctrl^C is pressed to terminate the program) and the program will determine using the NPDA functions whether this string is accepted or rejected by the NPDA. The program should also display the sequence of all the instantaneous descriptions in the format shown in the sample execution. In case the input string contains an invalid symbol, the program should stop processing the rest of the input. Note that you will need to use recursive backtracking to process an input string through the NPDA:

```
bool process( )
   if no more input symbol to read and the current state is a final state
      return true
   for each possible next state
      if process() = true
         return true
   return false
```

Deliverables:

- You will need to submit the C++ source file(s) in Blackboard by 04/28/24.
- You will need to demonstrate the project by 05/03/24.

Instructions:

- You MUST develop your program in the Linux environment using the g++ compiler.
- Write your driver code in *program2.cpp* and the NPDA class in *npda,h* and *npda.cpp*.
- Each program file should have the appropriate comment block at the top:

```
// Name: Your Name
// File Name:
// Date: Day Month, Year
// Description: Brief description of the code written in the file
```

- Use meaningful identifiers, sufficient and helpful comments, and a consistent coding style.
- This will be an individual project. However, feel free to contact the instructor (and ONLY the instructor) if you need help. You may discuss general aspects of the project with your classmates, but you may not collaborate in any way in producing code. Failure to follow these requirements will result in a grade of F in the project.

Grading:

This assignment will be graded as follows:

Program Correctness	90 pts
Program Style	10 pts

Sample Executions:

```
$ ./prog2
usage: <prog_name> <file_name>

$ ./prog2 foo
Automata file could not be opened!

$ ./prog2 test4.txt
---------N P D A---------
<states>
q0 q1 q2 q3
<alphabet>
a b
<stack alphabet>
0 1
<transitions>
(q0,*,0)->(q3,*)
(q0,a,0)->(q1,10)
(q1,a,1)->(q1,11)
(q1,b,1)->(q2,*)
(q2,*,0)->(q3,*)
(q2,b,1)->(q2,*)
<initial state>
q0
<stack start>
0
<final states>
q3
-------------------------

Enter a string to process (Ctrl^C to end): aabb

(q0,aabb,0)
|- (q1,abb,10)
|- (q1,bb,110)
|- (q2,b,10)
|- (q2,*,0)
|- (q3,*,*)
Accepted

Enter a string to process (Ctrl^C to end): aaa

Rejected

Enter a string (Ctrl^C to end):

(q0,*,0)
|- (q3,*,*)
Accepted
```

```
$ ./prog2 test5.txt
---------N P D A---------
<states>
q0 q1 q2
<alphabet>
a b
<stack alphabet>
a b z
<transitions>
(q0,a,a)->(q0,aa)
(q0,b,a)->(q0,ba)
(q0,a,b)->(q0,ab)
(q0,b,b)->(q0,bb)
(q0,a,z)->(q0,az)
(q0,b,z)->(q0,bz)
(q0,*,a)->(q1,a)
(q0,*,b)->(q1,b)
(q1,a,a)->(q1,*)
(q1,b,b)->(q1,*)
(q1,*,z)->(q2,z)
<initial state>
q0
<stack start>
z
<final states>
q2
-------------------------

Enter a string to process (Ctrl^C to end): abba

(q0,abba,z)
|- (q0,bba,az)
|- (q0,ba,baz)
|- (q1,ba,baz)
|- (q1,a,az)
|- (q1,*,z)
|- (q2,*,z)
Accepted

Enter a string to process (Ctrl^C to end): aba

Rejected

Enter a string (Ctrl^C to end):

Rejected
```