Hacettepe University
Computer Science and Engineering Department

Name and Surname	**:** Oğuzhan Ertekin
Identity Number	**:** 21946113
Course	**:** BBM203
Experiment	**:** Assignment 3
Subject	**:** DPDA Automata and Stacks
Data Due	**:** 17.12.2021
e-mail	**:** oguzhanertekin@hacettepe.edu.tr
Main Program	**:** src/Main.cpp

2. Software Using Documentation
2.1. Software Usage

This program is written in C++ language . It is compiled and executed in the C++ compiler. Input files are "dpda" and "dpda-input". In dpda file, there is a line starts with "Q" refers to states. There is a line starts with "A" refers to input alphabet. There is a line starts with "Z" refers to stack alphabet and there are line/lines start with "T" refers to transition rules.
In dpda-input file there are input symbols seperated with comma. There is also one output file where the correct transition rules and stack are printed.

The program is compiled as follows:

```
g++ -o Main *.cpp -std=c++11
```

The program is run as follows:

```
./Main   DPDA File    DPDA Input File    Output File
```

2.3 Error Messages
[Error 1: DPDA description is invalid!] :

Error Conditions:

- In DPDA input file, If there is a symbol that should not be in the input alphabet,
- In DPDA file, In transition rules lines, If the input symbol is not in the input alphabet or if the next state is not in the states,

Fixing Errors:

The content of the DPDA and DPDA Input Files should be rewritten in the desired correct format.

3.Software Design Notes
3.1. Desctiption of the program
3.1.1. Problem
We are expected to code an DPDA Automata Simulator using Stacks.

3.1.2.Solution
I used Stack structure fort he DPDA Simulator because in DPDA there is LIFO method. We were asked to check the compatibility of the inputs given in the DPDA Input File with the transition rules in the DPDA file, so I mostly used "for" and "while loops to check each transition rule. I also wrote functions to read input files, separate the lines according to a certain token, and store these lines in a vector. We needed top rint the correct transition rules and stack to the output file. I created an another function for this.

(Details are in title 3.3)

3.3. Main Data Structures
There is a stack data structure named "Stack" and we use push/pop operators to add or remove symbols from this stack. The function called "readFile" reads the input files and adds them to the vector in the desired format. The "split" function splits the string given as a parameter according to the desired delimeter. The "print_stack" function prints the stack elements side by side. The "print_output" function prints the desired output format in the output file.

3.4. Algorithm

1. Read DPDA and DPDA Input files line by line and create Output file.
   1.1. Create seperate vectors according to the initials of the lines in the DPDA file
   (For Q: states, For A: input alphabet, For Z: stack alphabet, For T: transitions)
   l.2. Store input symbols in DPDA Input file in vector.
   1.3 If there is a symbol that should not be in the input alphabet or if there is a state that should not be in the states; Print error message.
   1.4. Create stack

2. For every transitions:
   2.1. Check if the start state is equals to current state
   2.2. Check if the input symbol is equals to current symbol.
   2.3. Check if the pop element is equals to top element of the stack
   2.4. If these three conditions are met:
      2.4.1. Pop “top element” from stack
      2.4.2. Push given element to stack.
      2.4.2. Set current state to next state.

3. If all input symbols are gone ,all transition rules are provided, stack is empty and last state is final state:
   3.1. Print “ACCEPT” to the output file.
4. If condition 3 is not met:
   4.1 Print “REJECT” to the output file.
5. Close files

4. SOFTWARE TESTING NOTES :

4.1. Bugs and Software Reliability:

There are a few bugs in this assignment. Most common bug’s message is “terminate called after throwing an instance of 'std::bad_alloc' what(): std::bad_alloc”. This error was encountered when vectors limit is exceeded or memory is not well used. Used bool keys and if/else status checking in some parts of the code to solve this bug. Another bug was “Segmentation Error” while compiling the code on “dev server”. It was solved by creating new input files on the “dev server” and copying the texts written in the input files into it.

REFERENCES

1-BBM 201 Lecture Slides
2-www.geeksforgeeks.org
3-www.cplusplus.com
4-www.stackoverflow.com
5-www.learncpp.com
6- www.w3schools.com
7-www.tutorialspoint.com
8-www.codecademy.com
9-www.programiz.com