Assignment 3

Hacettepe University

Computer Science and Engineering Department

Name and Surname	**:** Mehmet Ertaş
Identity Number	**:** b2200765035
Course	**:** Artificial Engineering Course
Data Due	**:** 17/12/2021
e-mail	**:** mehertas20@gmail.com

2. Software Using Documentation

2.1. Software Usage

This software is used by inputting 3 text files while being booted up by the terminal. The first one gives different states, our alphabet and rules. The second one provides us with input and the third us the file for output which we will write on.

2.2. Error Messages

This software offers 2 different error messages.

[1]: Error [1]: DPDA description is invalid!
This error would happen if you have problem with your DPDA file such as:

- Initial state or final state isn't in the all states

- You entered a character that is not in alphabet

[2]: Error[2]: Input is invalid!
This error would happen if you entered a character in the input file that is not in alphabet that is not stated in the DPDA file

3. Software Design Notes

3.1. Description of the Program

3.1.1. Problem

In this assignment we were assigned to make a program which would take a file including rules and a file including input to create a file including output based on the rules which was established in the first file.

3.1.2. Solution

There are many ways to do solve this problem such as using vectors or arrays. While vectors are great because they are deterministic and arrays allow us to limit how many things we can hold at once they're bad at keeping track of where everything is. To answer that I've used Queue and Stack. Not only we can store as many data as we need but we can also choose which one goes out first.

3.2. Algorithm

```
1. Initialization
    1.1 Initialize data space
    1.2 Start reading DPDA file
        1.2.1 Read and save STATE, ALPHABET and STACK ALPHABET data
        1.2.2 Read and check RULES data, then save if successful
    1.3 Start reading INPUT file
        1.3.1 Read and INPUT data, then save if successful
2. Transitioning
    2.1 Start reading RULES data
        2.1 Activate the rules if STATES, READ and POP data match
        2.2 Print out a part of the OUTPUT
3. Finishing
    3.1 Check if RULES data is empty if it is empty also check INPUT data
        3.1.1 Check if the final state is in accept position and STACK is empty
    3.2 If everything empty as well as in the accept position print ACCEPT otherwise print
```

4. Software Testing Notes

4.1 Bugs and Software Reliability

As far as I know my program doesn’t have any bugs

REJECT