2. Software Using Documentation

2.1. Software Usage

Program reads inputs from files which are given as command-line arguments to program. Then gives output as a file which has the indicated name from command line. After reading and storing inputs in, it process as asked in project. Then saves results as file. After execution of program we see success message in terminal.

2.3 Error Messages

If input files are invalid then we'll see in output files as 'Error [1]:DPDA description is invalid!'

This error happens if there is discrepancy between Q, A, Z.

3.Software Design Notes

3.1. Desctiption of the program

3.1.1. Problem

Problem includes using DPDA. The program accepts a string belonging to a language or rejects a string not belonging to the language. The program is to read a description of a DPDA and one input for it. It will process an input and output the execution path. After processing one string, the program should reset the automata (this includes emptying the stack if necessary) and proceed with the next string.

3.1.2.Solution

We take inputs and process them then give output as accepted or rejected with steps. First line of dpda file gives us state names and starting-ending states. And at problem description it says that we will use the final state as acceptance criteria in the experiment.
If we look closer, we'll see that DPDA has a logic. At states we have informations about state id, symbols which is for input string, element for pop from stack, next state id, element for push to stack. And these are enough for my solution. Solution is simply like that:
First checking if variables are valid. If not write error message to file then exit to next input. If valid we'll contunie with comparing. At every compare we'll choose from states(T) and contuine with it for that loop. We should loop until input string achieved and ended with Q we want or until no possible solutions. It looks properties that they have for every T for next process. And contunie to the end. Comparing factors are better explained in algorithm section and in code with command lines.

3.4. Algorithm

1. Make initialisation for data in files by reading them. Mostly string vectors are used for that.
2. Check if data are valid
3. Loop for every string
    3.1 Look for state id suitability
    3.2. Look for string order
    3.3. Look for element which will be popped from stack
    3.4. Process for remaning T and write to next line at file with current stack
4. Write acceptance

Software Testing Notes *:*

It successed every test cases.

4. SOFTWARE TESTING NOTES :

4.1. Bugs and Software Reliability

I tested, debugged for different cases. I think I covered all bugs.

REFERENCES