Samantha Rack
CSE 30151 – Theory of Computing
Project 2

Language and Platform:

The language in which I developed this project is C++ (same as project 1). The program was developed on one of the student machines, so the platform was Linux Redhat.

High Level Description:

The simulation program begins by creating a deterministic PDA object from the description file. When this object is created, all transitions out of each state are compared pairwise to determine if any of the rules of deterministic push-down automata are violated by these transitions. After this initialization, the program reads lines of inputs, and then the machine processes each of the strings of inputs received, resetting after each string.

The major data structures in the PDA class hold each of the components of the machine. The alphabets, set of states, and set of accept states are stored in arrays of strings. The stack of the machine is stored in a stack structure. Finally, the transitions of each state are stored in an array.

On each input, the program loops through the transitions for the current state, searching for a match to the input character. If one is found, then that transition's stack read is checked for "e", indicating that nothing has to be read off the stack, or for a match with the top of the stack. On a match with either of those, the state advances to the specified state. On reset of the machine and after each input is read, empty string input transitions are checked. The same process is followed as for an input character, but the match of the transition's input is "e". The stack read for a found transition could match either "e" (where the stack would not be popped) or the top of the stack (where the stack would be popped). Until no transition match is found for a given state, the empty string transition is checked. Then the next input character is read.

Test Programs and Inputs:

To verify the correctness of the simulation, I tested the program with all of the given test PDAs (2 deterministic with input/output examples, and 2 nondeterministic PDAs that gave an error).

Difficulties and Implementation Decisions:

The main difficulty in developing this program was the transition function. I chose to not represent the transition function as I had for the previous assignment (3D vector with a list of next states for every current state, input pair; for this assignment it would have been a 3D vector with a next state for every current state, input, stack read triple) because it would be a sparse data structure. Therefore, I chose to only save a list of transitions for each state, and to iterate through these for each transition. This is more computationally-costly, but because of the limited size of the machines to be simulated with this program, it was effective for this project.