**Instructions:**

You can run this program on shell buy typing “clisp –i 2PDA.lsp” in whatever directory you downloaded it to. Once running you can type (make-pda “filename”) and it will read in the file and create the appropriate data structure. After you have created the pda you can use the (test-string “string” fast) function. Replace fast with either a “t” or NIL depending on whether you want a fast run or not.

If you need to clear any of the data structures at any point during the program you can run (clear-pda) to clear all the lists related to the actual machine and (clear-stacks) to clear the stacks.

If you want to know what is in the pda the (print-stuff) function will print the value of all of the global variables in the program.

At the bottom of the source code are three lines that show how to build the example 1 pda and test a string with it.

**Notes:**

I changed the implementation of the file format for the pdas to denote what character is being read in for which transaction. It was added after the spot denoting what the new state will be. The syntax for one transition is now:

New state, scanned character, pop stack 1, push stack 1, pop stack 2, push stack 2

The three examples provided have been changed to this syntax and are included with this document and the program.

When nil is passed for the fast argument of the test-string it will print: the contents of both stacks, the current state after the transition was taken, the character just processed, and the transition that was taken.

You may see the string “String Accepted” printed twice at the end of call to test-string. This is because a lisp function returns the value of the last evaluated expression and I couldn’t figure out to have it return nil.

Characters in lisp when not properly formatted look like #\c.

If you need me to demo the program just send me an email I’ll be here all week and I only have one final on Monday and two on Thursday.