Milestones Challenge 7
Name: Paul Blair
NetID: VBQ669

Rubric:

+2   Code is well formatted, commented (inc. name, assignment, and
overview), with reasonable variable names
+4   Passes memory tests
+4   Properly read input from the user
+10   Properly parse/tokenize user input to be able to implement file
redirection and piping
+10   Properly utilize fork() and execvp() to spawn child process and
execute commands with proper error handling
+10   Passes all test cases (these will demonstrate the full
capability of the shell)

Link to Repository:

  https://github.com/blairpaul8/myshell_cs302

Timelog:

  – 30 minutes writing a Makefile to make compiling my code easier.
  – 2 hours working on learing how to read input in C and implemented
my getline() function.
  – 3 hours learning how to parse input with strtok() and how to use
fork() and execvp().

    Note:
      in my git repo there are example programs I wrote while I was
learning about fork() and
      process id's.

Summary:

    So far on this final project I have implemented my own getline
function to read user input from from stdin by utilizing getchar().
Additionally, I spent time reading through Dr. Planks lecture notes on
fork, wait, and exec to learn how to spawn sub processes and execute
commands given by the user. While learning this I wrote the example
problems Dr. Plank gives in his lectures. After this I was able to
successfully write my execute() function to create a child process and
use execvp() to execute the commands stored in the args array. Lastly,
I also wrote a Makefile for facilitate compiling my code which will
also have an option to run make test to run the test cases in the
future.

To finish this project I still need to implement file redirection and piping. Also I will be writing an implementation of a hash map to store the names of the built-in commands (cd, echo, test, etc) to provide constant time look up of these commands if one of these commands is given by the user. Lastly, I need to add error checking to things like fork() or execvp() in the case one of these systems calls fails.