CIS 212: Project #1C
Assigned: Monday November 5th, 2018
Due Monday, November 12th, 2018
(which means submitted by 6am on November 13th, 2018)
Worth 3% of your grade

Please read this entire prompt!

1) Download the file passwd_212 from the course website. Also download the files correct_output_passwd_212, correct_output_proj1c, and check1c. Make sure to give check1c and your script proj1c.sh execute permissions.
2) Write a script that accomplishes two goals
   a. Removes user IDs in the 90s (i.e., _tokend, _securityagent, … _unknown). You must do this using "head" and "tail" commands. You must also automate finding the line numbers to cut out. For example, assume it is fair game for me to add or subtract lines from the file – the input I test on may be slightly different than passwd_212. That said, you may assume that the first user ID to cut is 91 and the last user ID to cut is 99. Hint 1: there is a version of grep called "fgrep" which will look at a file. If you call fgrep with the "-n" flag, then it will print line numbers. Hint 2: I did this by creating two files (everything before 91 and everything after 99) and then combining the two files. Call the modified version of this file "passwd_212_no90s".
   b. Counts how many people use each shell type. Note: this can be accomplished with a single command (and a lot of pipes) if you use "sort" and "uniq". Also, I found a spurious line of whitespace. I got rid of that from my report by piping the output to "grep /" at the end. I also used "grep –v" to get rid of comments. Note that the shell wants to interpret # as a comment to the shell … you need to quote it '#' so the shell realizes you are passing the character as an argument, and not issuing something the shell itself should be processing.
      NOTE: Mac outputs different whitespace than VirtualBox. Must run this on VirtualBox…

How to test?: use the check_1c script
What to upload? Your script. It should have the name proj1c.sh

Lastly: the simplest way to do this is to experiment by running the commands in the Unix shell. Once you find the right commands, then copy them to the script.

Other notes:
LINES=$(wc –l file1)
is a useful command. wc –l finds the number of lines in file1. $(…) invokes the command. Overall, this would define a new variable called LINES that would be equal to invoking the command "wc –l file1" in the shell. Of course, $(…) is useful in other contexts too.