CSc 3320: Systems Programming

Fall 2021

Midterm 1: Total points = 100

Submission instructions:

1. Create a Google doc for your submission.
2. Start your responses from page 2 of the document and copy these instructions on page 1.
3. Fill in your name, campus ID and panther # in the fields provided. If this information is missing TWO POINTS WILL BE DEDUCTED.
4. Keep this page 1 intact. If this *submissions instructions* page is missing in your submission TWO POINTS WILL BE DEDUCTED.
5. Start your responses to each QUESTION on a new page.
6. If you are being asked to write code copy the code into a separate txt file and submit that as well. The code should be executable. E.g. if asked for a C program then provide myfile.c so that we can execute that script. In your answer to the specific question, provide the steps on how to execute your file (like a ReadMe).
7. If you are being asked to test code or run specific commands or scripts, provide the evidence of your outputs through a screenshot and/or screen video-recordings and copy the same into the document.
8. Upon completion, download a .PDF version of the google doc document and submit the same along with all the supplementary files (videos, pictures, scripts etc).
9. Scripts/Code without proper comments, indentation and titles (must have the name of the program, and name & email of the programmer on top the script).

Full Name: Talaal Tahir

Campus ID: Ttahir1

Panther #: 002504147

Question 1

(20 pts) Pick any of your 10 favourite unix commands. For each command run the *man* command and copy the text that is printed into a mandatabase.txt. Write a shell script *helpme.sh* that will ask the user to type in a command and then print the manual's text associated with that corresponding command. If the command the user types is not in the database then the script must print *sorry, I cannot help you*

1. Step 1: Create a mandatabase.txt

    a. **vi mandatabase.txt**

```
[ttahir1@gsuad.gsu.edu@snowball ~]$ vi mandatabase.txt
[ttahir1@gsuad.gsu.edu@snowball ~]$ ls
a.out          foo.class  hello     helpme.sh     Lab2_P2  mandatabase.txt  myName     pdf_files.tar     sh_files
checkError.sh  foo.java   hello.c   homework.pdf  Lab3     midterm          myName.c   pdf_sh_files.tar  sh_files.tar
csc3320        foo.sh     hello.sh  homeworks     Lab4     myexamfile.txt   pdf_files  public            simple.sh
[ttahir1@gsuad.gsu.edu@snowball ~]$
```

2. Step 2: Add 10 unix commands to mandatabase.txt

    a. **man #command >> mandatabse.txt**

```
[[ttahir1@gsuad.gsu.edu@snowball ~]$ man ls >> mandatabase.txt
[[ttahir1@gsuad.gsu.edu@snowball ~]$ man cd >> mandatabase.txt
[[ttahir1@gsuad.gsu.edu@snowball ~]$ man grep >> mandatabase.txt
[[ttahir1@gsuad.gsu.edu@snowball ~]$ man vi >> mandatabase.txt
[[ttahir1@gsuad.gsu.edu@snowball ~]$ man sed >> mandatabase.txt
[[ttahir1@gsuad.gsu.edu@snowball ~]$ man awk >> mandatabase.txt
[[ttahir1@gsuad.gsu.edu@snowball ~]$ man sort >> mandatabase.txt
[[ttahir1@gsuad.gsu.edu@snowball ~]$ man mkdir >> mandatabase.txt
[[ttahir1@gsuad.gsu.edu@snowball ~]$ man wc >> mandatabase.txt
[[ttahir1@gsuad.gsu.edu@snowball ~]$ man cat >> mandatabase.txt
[[ttahir1@gsuad.gsu.edu@snowball ~]$ vi mandatabse.txt
[[ttahir1@gsuad.gsu.edu@snowball ~]$ rm mandatabse.txt
[[ttahir1@gsuad.gsu.edu@snowball ~]$ vi mandatabase.txt
```

```
LS(1)                                   User Commands                                   LS(1)


NAME
       ls - list directory contents

SYNOPSIS
       ls [OPTION]... [FILE]...

DESCRIPTION
       List  information about the FILEs (the current directory by default).  Sort entries alphabetically if none of -cftuvSUX
       nor --sort is specified.

       Mandatory arguments to long options are mandatory for short options too.

       -a, --all
              do not ignore entries starting with .

       -A, --almost-all
              do not list implied . and ..

       --author
              with -l, print the author of each file

       -b, --escape
              print C-style escapes for nongraphic characters

       --block-size=SIZE
              scale sizes by SIZE before printing them; e.g., '--block-size=M' prints sizes in units of 1,048,576  bytes;  see
              SIZE format below

       -B, --ignore-backups
              do not list implied entries ending with ~

       -c     with  -lt:  sort by, and show, ctime (time of last modification of file status information); with -l: show ctime
              and sort by name; otherwise: sort by ctime, newest first

       -C     list entries by columns

       --color[=WHEN]
              colorize the output; WHEN can be 'never', 'auto', or 'always' (the default); more info below
```

3. Step 3: Write Shell Script helpme.sh that prints desired text from mandatabse.txt

```
[ttahir1@gsuad.gsu.edu@snowball ~]$ vi helpme.sh
[ttahir1@gsuad.gsu.edu@snowball ~]$ chmod a+x helpme.sh
[ttahir1@gsuad.gsu.edu@snowball ~]$ ./helpme.sh
```

```
#!/bin/bash
# Talaal Tahir
# TTahir1@student.gsu.edu
# Find command in mandatabase.txt and print it

echo "Type command that you are looking for:"
read cmd

if [ $cmd == ls ] || [ $cmd == LS ] || [$cmd == Ls ]
then
sed -n '/LS(1)/,/LS(1)/p' mandatabase.txt

elif [ $cmd == cd ] || [$cmd == CD] || [ $cmd == Cd ]
then
sed -n '/CD(1)/,/CD(1)/p' mandatabase.txt

elif [ $cmd == grep ] || [ $cmd == GREP ] || [ $cmd == Grep ]
then
sed -n '/GREP(1)/,/GREP(1)/p' mandatabase.txt

elif [ $cmd == vi ] || [ $cmd == VI ] || [ $cmd == Vi ]
then
sed -n '/VI(1)/,/VI(1)/p' mandatabase.txt

elif [ $cmd == sed ] || [ $cmd == SED ] || [ $cmd == Sed ]
then
sed -n '/SED(1)/,/SED(1)/p' mandatabase.txt

elif [ $cmd == awk ] || [ $cmd == AWK ] || [ $cmd == Awk ]
then
sed -n '/AWK(1)/,/AWK(1)/p' mandatabase.txt

elif [ $cmd == sort ] || [ $cmd == SORT ] || [ $cmd == Sort ]
then
sed -n '/SORT(1)/,/SORT(1)/p' mandatabase.txt

elif [ $cmd == mkdir ] || [ $cmd == MKDIR ] || [ $cmd == Mkdir ]
then
sed -n '/MKDIR(1)/,/MKDIR(1)/p' mandatabase.txt

elif [ $cmd == wc ] || [ $cmd == WC ] || [ $cmd == Wc ]
then
sed -n '/WC(1)/,/WC(1)/p' mandatabase.txt

elif [ $cmd == cat ] || [ $cmd == CAT ] || [ $cmd == Cat ]
then
sed -n '/CAT(1)/,/CAT(1)/p' mandatabase.txt

else
echo "sorry, I cannot help you"

fi
```

```
[ttahir1@gsuad.gsu.edu@snowball ~]$ ./helpme.sh
Type command that you are looking for:
wc
WC(1)                                                   User Commands                                                   WC(1)



NAME
       wc - print newline, word, and byte counts for each file

SYNOPSIS
       wc [OPTION]... [FILE]...
       wc [OPTION]... --files0-from=F

DESCRIPTION
       Print newline, word, and byte counts for each FILE, and a total line if more than one FILE is specified.  With no FILE,
       or when FILE is -, read standard input.  A word is a non-zero-length sequence of characters delimited by  white  space.
       The options below may be used to select which counts are printed, always in the following order: newline, word, charac-
       ter, byte, maximum line length.

       -c, --bytes
              print the byte counts

       -m, --chars
              print the character counts

       -l, --lines
              print the newline counts

       --files0-from=F
              read input from the files specified by NUL-terminated names in file F; If F is - then read names  from  standard
              input

       -L, --max-line-length
              print the length of the longest line

       -w, --words
              print the word counts

       --help display this help and exit

       --version
              output version information and exit

       GNU  coreutils online help: <http://www.gnu.org/software/coreutils/> Report wc translation bugs to <http://translation-
       project.org/team/>

AUTHOR
       Written by Paul Rubin and David MacKenzie.

COPYRIGHT
       Copyright  ©  2013  Free  Software  Foundation,  Inc.   License  GPLv3+:  GNU  GPL  version  3  or  later
       <http://gnu.org/licenses/gpl.html>.
       This  is  free  software: you are free to change and redistribute it.  There is NO WARRANTY, to the extent permitted by
       law.

SEE ALSO
       The full documentation for wc is maintained as a Texinfo manual.  If the info and wc programs are properly installed at
       your site, the command

              info coreutils 'wc invocation'

       should give you access to the complete manual.



GNU coreutils 8.22                                      November 2020                                                   WC(1)
```

4. Step 4: If command is not in database print “sorry, I cannot help you”

```
[ttahir1@gsuad.gsu.edu@snowball ~]$ ./helpme.sh
Type command that you are looking for:
touch
sorry, I cannot help you
```

Question 2

(10pts each) On your computer open your favourite Wikipedia page. Copy the text from that page into a text file **myexamfile.txt** and then copy that file to a directory named **midterm** (use mkdir to create the directory if it doesn't exist) in your snowball server home directory.

a. Write a shell script that will find the number of statements in the text. A statement is defined as the collection of text between two periods (full-stops).

b. Update the script to present a tabular list that shows the number of words and number of letters in each statement.

1. Step 1: Open wikipedia page and create file myexamfile.txt with text from the page

    a. **vi myexamfile.txt**

```
[ttahir1@gsuad.gsu.edu@snowball ~]$ vi myexamfile.txt
```

```
Valorant (stylized as VALORANT) is a free-to-play first-person hero shooter developed and published by Riot Games, for Microsoft W
indows. First teased under the codename Project A in October 2019, the game began a closed beta period with limited access on Apri
l 7, 2020, followed by an official release on June 2, 2020. The development of the game started in 2014.
Gameplay
Valorant is a team-based first-person hero shooter set in the near future.[2][3][4][5] Players play as one of a set of agents, cha
racters designed based on several countries and cultures around the world.[5] In the main game mode, players are assigned to eithe
r the attacking or defending team with each team having five players on it. Agents have unique abilities, each requiring charges,
as well as a unique ultimate ability that requires charging through kills, deaths, orbs, or spike actions. Every player starts eac
h round with a "classic" pistol and one or more "signature ability" charges.[3] Other weapons and ability charges can be purchased
 using an in-game economic system that awards money based on the outcome of the previous round, any kills the player is responsibl
e for, and any actions taken with the spike. The game has an assortment of weapons including secondary guns like sidearms and prim
ary guns like submachine guns, shotguns, machine guns, assault rifles and sniper rifles.[6][7] There are automatic and semi-automa
tic weapons that each have a unique shooting pattern that has to be controlled by the player to be able to shoot accurately.[7] Di
fferent agents allow players to find more ways to plant the Spike and style on enemies with scrappers, strategists, and hunters of
 every description. It currently offers 16 agents to choose from.[8][9] They are Brimstone, Viper, Omen, Cypher, Sova, Sage, Phoen
ix, Jett, Raze, Breach, Reyna, Killjoy, Skye, Yoru, Astra, and KAY/O.
Unrated
In the standard non-ranked mode, the match is played as best of 25 - the first team to win 13 rounds wins the match. The attacking
 team has a bomb-type device called the Spike. They must deliver and activate the Spike on one of the multiple specified locations
 (bomb sites). If the attacking team successfully protects the activated Spike for 45 seconds it detonates, destroying everything
in a specific area, and they receive a point.[3] If the defending team can deactivate the spike, or the 100-second round timer exp
ires without the attacking team activating the spike, the defending team receives a point.[10] If all the members of a team are el
iminated before the spike is activated, or if all members of the defending team are eliminated after the spike is activated, the o
pposing team earns a point.[3] If both teams win 12 rounds, sudden death occurs, in which the winning team of that round wins the
match, differing from overtime for competitive matches. Additionally, if after 4 rounds, a team wishes to forfeit that match, they
 may request a vote to surrender. If the vote is unanimous, the winning team gets all the victory credit for every round needed to
 bring them to 13, with the forfeiting team receiving losing credit.[11] A team gets only two chances to surrender: one as the att
ackers and the other as the defenders.
Spike Rush
In the Spike Rush mode, the match is played as best of 7 rounds - the first team to win 4 rounds wins the match. Players begin the
 round with all abilities fully charged except their ultimate, which charges twice as fast as in standard games. All players on th
e attacking team carry a spike, but only one spike may be activated per round. Guns are randomized in every round and every player
 begins with the same gun. Ultimate point orbs in the standard game are present, but there are multiple power-up orbs instead.[12]
Competitive
Competitive matches are the same as unranked matches with the addition of a win-based ranking system that assigns a rank to each p
layer after 5 games are played. Before you can play in competitive games, you will need to win 10 unrated matches beforehand.[13]
In July 2020, Riot introduced a "win by two" condition for competitive matches, where instead of playing a single sudden death rou
nd at 12-12, teams will alternate playing rounds on attack and defense in overtime until a team claims victory by securing a two-m
atch lead. Each overtime round gives players the same amount of money to purchase guns and abilities, as well as approximately hal
f of their ultimate ability charge. After each group of two rounds, players may vote to end the game in a draw, requiring 6 player
s after the first set, 3 after the second, and thereafter only 1 player to agree to a draw. The competitive ranking system ranges
from iron to radiant. Every rank but Immortal and Radiant has 3 tiers.[14][12] Immortal & Radiant are reserved for top 500 players
 in which there is a number associated to their rank allowing players in the top 500 to have a metric in which they can compare ho
w they rank up to others at their level.[15]
Deathmatch
The Deathmatch mode was introduced on August 5, 2020.[16] 14 players enter a 9-minute free-for-all match and the first person to r
each 40 kills or the player who has the most kills when time is up wins the match. Players spawn in with a random agent, and all a
bilities are disabled during the match which indulges pure gunplay. Green health packs drop on every kill, which set the player at
 maximum health, armor, and ammunition, unless the player is using a machine gun, which only gives the player an additional 30 bul
lets.[17]
```

2. Step 2: Create directory midterm and copy myexamfile.txt there

    a. **mkdir midterm**

    b. **cp myexamfile.txt ~/midterm**

```
[ttahir1@gsuad.gsu.edu@snowball ~]$ mkdir midterm
[ttahir1@gsuad.gsu.edu@snowball ~]$ ls
a.out          foo.java  hello.sh      Lab2_P2          mandatabse.txt  myName.c          public
checkError.sh  foo.sh    helpme.sh     Lab3             midterm         pdf_files         sh_files
csc3320        hello     homework.pdf  Lab4             myexamfile.txt  pdf_files.tar     sh_files.tar
foo.class      hello.c   homeworks     mandatabase.txt  myName          pdf_sh_files.tar  simple.sh
[ttahir1@gsuad.gsu.edu@snowball ~]$ cp myexamfile.txt ~/midterm
[ttahir1@gsuad.gsu.edu@snowball ~]$ cd midterm
[ttahir1@gsuad.gsu.edu@snowball midterm]$ ls
myexamfile.txt
```

3. Step 3: Create shell script to find a number of statements

    a. **vi search.sh**

```
[[ttahir1@gsuad.gsu.edu@snowball midterm]$ vi search.sh
[[ttahir1@gsuad.gsu.edu@snowball midterm]$ chmod a+x search.sh
[[ttahir1@gsuad.gsu.edu@snowball midterm]$ ./search.sh
Number of Statements in myexamfile.txt is:
57
```

```
#!/bin/bash
# Talaal Tahir
# Ttahir1@student.gsu.edu
# Serach through myexamfile.txt to find number of statements

echo "Number of Statements in myexamfile.txt is:"

sed 's/\../\n/g' myexamfile.txt > temp.txt
cat temp.txt | wc -l
```

4. Step 4: Create a tabular list that shows the number of words and letters in each statement.

```
#!/bin/bash
# Talaal Tahir
# Ttahir1@student.gsu.edu
# Serach through myexamfile.txt to find number of statements

echo "Number of Statements in myexamfile.txt is:"

sed 's/\../\n/g' myexamfile.txt > temp.txt
cat temp.txt | wc -l

echo "Number of words and characters in each Statements is:"
echo "----------------------"
while read line
do
echo "$line" | wc -w -c
echo "----------------------"
done < temp.txt
```

```
Number of words and characters in each Statements is:
----------------------
     19     137
----------------------
     33     181
----------------------
      8      44
----------------------
     12      74
----------------------
     21     129
----------------------
     24     130
----------------------
     24     160
----------------------
     16      99
----------------------
     39     227
----------------------
     26     173
----------------------
     28     165
----------------------
     25     154
----------------------
      8      45
----------------------
     20     133
----------------------
     24     116
----------------------
     10      59
----------------------
     16      97
----------------------
     25     160
----------------------
     28     174
----------------------
     36     196
----------------------
     27     157
----------------------
     19     107
----------------------
     29     164
----------------------
     19      99
----------------------
     25     112
----------------------
     21     129
----------------------
     18      95
----------------------
     14      77
----------------------
     16      99
----------------------
     29     165
----------------------
     16      93
----------------------
     46     274
----------------------
     25     154
----------------------
     37     185
----------------------
      9      59
----------------------
      9      48
----------------------
     44     229
----------------------
     10      57
----------------------
     31     158
----------------------
     19     114
----------------------
     34     197
----------------------
     38     230
----------------------
     13      68
----------------------
     33     179
----------------------
     31     168
----------------------
     23     126
----------------------
     22     133
----------------------
     18     109
----------------------
     15      94
----------------------
     30     168
----------------------
     11      52
----------------------
      8      51
----------------------
     16      76
----------------------
     15      85
----------------------
     14      82
----------------------
     13      78
----------------------
     11      50
----------------------
```

Question 3

(20pts) Design a calculator using a shell script using regular expressions. The calculator, at the minimum, must be able to process addition, subtraction, multiplication, division and modulo operations. It must also have cancel and clear features.

1. Step 1: create shell script file calculator.sh

    a. **vi calculator.sh**

    b. **chmod a+x calculator.sh**

```
[[ttahir1@gsuad.gsu.edu@snowball midterm]$ vi calculator.sh
[[ttahir1@gsuad.gsu.edu@snowball midterm]$ chmod a+x calculator.sh
```

2. Step 2: Write Shell script to do calculator

```
[[ttahir1@gsuad.gsu.edu@snowball midterm]$ ./calculator.sh
Welcome to the Calcultor program!
-----------------------------
Please enter an expression or cancel to exit:

-------------------------
[2+2
2+2 = 4
Would you like to add to the previous sum: Type yes or no
[yes
Please enter an expression or cancel to exit:

-------------------------
[2+2
4+2+2 = 8
Would you like to add to the previous sum: Type yes or no
[yes
Please enter an expression or cancel to exit:

-------------------------
[4*2
8+4*2 = 16
Would you like to add to the previous sum: Type yes or no
[no
[ttahir1@gsuad.gsu.edu@snowball midterm]$
```

```
#!/bin/bash
# Talaal Tahir
# Ttahir1@studnet.gsu.edu

echo "Welcome to the Calcultor program!"
echo "--------------------------------"
awnser="yes"
prevSum=0
totalSum=0
prevTotal=0
while [ $awnser == "yes" ] || [ $awnser == "YES" ] || [ $awnser == "Yes" ]
do
echo "Please enter an expression or cancel to exit:"
echo "-----------------------------------------"
read expression

#Addition problems
if [[ $expression =~ [0-9]*\+[0-9]* ]]
then
currSum=$(echo $expression | awk -F+ '{print ($1+$2)}')
totalSum=`echo "scale=2; $currSum + $prevSum" | bc`
if [ $prevTotal == 0 ]
then
echo $expression "=" $totalSum
else
echo $prevTotal"+"$expression "=" $totalSum
fi

#Subtraction expressions
elif [[ $expression =~ [0-9]*-[0-9]* ]]
then
currSum=$(echo $expression | awk -F- '{print ($1-$2)}' )
totalSum=`echo "scale=2; $currSum + $prevSum" | bc`
if [ $prevTotal == 0 ]
then
echo $expression "=" $totalSum
else
echo $prevTotal"+"$expression "=" $totalSum
fi

#Multiplication expressions
elif [[ $expression =~ [0-9]*\*[0-9]* ]]
then
currSum=$(echo $expression | awk -F* '{print ($1*$2)}' )
totalSum=`echo "scale=2; $currSum + $prevSum" | bc`
if [ $prevTotal == 0 ]
then
echo $expression "=" $totalSum
else
echo $prevTotal"+"$expression "=" $totalSum
fi
```

```
#Division expressions
elif [[ $expression =~ [0-9]*/[0-9]* ]]
then
currSum=$(echo $expression | awk -F/ '{print ($1/$2)}' )
totalSum=`echo "scale=2; $currSum + $prevSum" | bc`
if [ $prevTotal == 0 ]
then
echo $expression "=" $totalSum
else
echo $prevTotal"+"$expression "=" $totalSum
fi

#Mod expressions
elif [[ $expression =~ [0-9]*%[0-9]* ]]
then
currSum=$(echo $expression | awk -F% '{print ($1%$2)}' )
totalSum=`echo "scale=2; $currSum + $prevSum" | bc`
if [ $prevTotal == 0 ]
then
echo $expression "=" $totalSum
else
echo $prevTotal"+"$expression "=" $totalSum
fi

#Clear the previous sum and total
elif [ $expression = "clear" ] || [ $expression == "CLEAR" ] || [ $expression == "Clear" ]
then
prevSum=0
prevTotal=0
echo "Previous sum has been cleared"

elif [ $expression = "cancel" ] || [ $expression == "CANCEL" ] || [ $expression == "Cancel" ]
then
echo "Calculator program has ended"
break
fi

#Check if user wants to continue
echo "Would you like to add to the previous sum: Type yes or no"
read awnser

#If user wants to continue set sums so they can continue to add to the total
if [ $awnser == "yes" ] || [ $awnser == "YES" ] || [ $awnser == "Yes" ]
then
prevSum=$totalSum
currSum=0
prevTotal=$totalSum

else
break

fi
done
```

Question 4

(20pts) Build a phone-book utility that allows you to access and modify an alphabetical list of names, addresses and telephone numbers. Use utilities such as awk and sed, to maintain and edit the file of phone-book information. The user (in this case, you) must be able to read, edit, and delete the phone book contents. The permissions for the phone book database must be such that it is inaccessible to anybody other than you (the user).

1. Step 1: Create a text file for the phone book and a shell script file

    a. **vi phonebook.txt**

    b. **vi phonesearch.sh**

```
[[ttahir1@gsuad.gsu.edu@snowball midterm]$ vi phonebook.txt
[[ttahir1@gsuad.gsu.edu@snowball midterm]$ vi phonesearch.sh
[[ttahir1@gsuad.gsu.edu@snowball midterm]$ chmod a+x phonesearch.sh
[[ttahir1@gsuad.gsu.edu@snowball midterm]$ chmod 700 phonebook.txt
```

2. Step 2: Write shell script for phone search utility

```
Phone Book Utility
------------------
1: Display Book
2: Add Person
3: Edit Person
4: Find Person
5: Delete Person
6: Exit

Enter [1-6]
1
phonebook is empty
Phone Book Utility
------------------
1: Display Book
2: Add Person
3: Edit Person
4: Find Person
5: Delete Person
6: Exit

Enter [1-6]
2
Enter First,Last Name, Number, and address
First Name:
Talaal
Last Name:
Tahir
Number:
40409812345
Address:
1490 Waverly glen
New info added
```

```
Phone Book Utility
------------------
1: Display Book
2: Add Person
3: Edit Person
4: Find Person
5: Delete Person
6: Exit

Enter [1-6]
2
Enter First,Last Name, Number, and address
First Name:
Bob
Last Name:
Marley
Number:
1800789765
Address:
210 Pidemont
New info added
Phone Book Utility
------------------
1: Display Book
2: Add Person
3: Edit Person
4: Find Person
5: Delete Person
6: Exit

Enter [1-6]
1
First Name: Bob , Last Name: Marley ,  Number: 1800789765 , Address: 210 Pidemont
First Name: Talaal , Last Name: Tahir ,  Number: 40409812345 , Address: 1490 Waverly glen
```

```
Phone Book Utility
------------------
1: Display Book
2: Add Person
3: Edit Person
4: Find Person
5: Delete Person
6: Exit

Enter [1-6]
3
Who would you like to edit?
[Talaal
First Name: Talaal , Last Name: Tahir ,  Number: 40409812345 , Address: 1490 Waverly glen
What would you like to edit
[Talaal
What would you like to edit it to?
[John
Phone Book Utility
------------------
1: Display Book
2: Add Person
3: Edit Person
4: Find Person
5: Delete Person
6: Exit

Enter [1-6]
1
First Name: Bob , Last Name: Marley ,  Number: 1800789765 , Address: 210 Pidemont
First Name: John , Last Name: Tahir ,  Number: 40409812345 , Address: 1490 Waverly glen
```

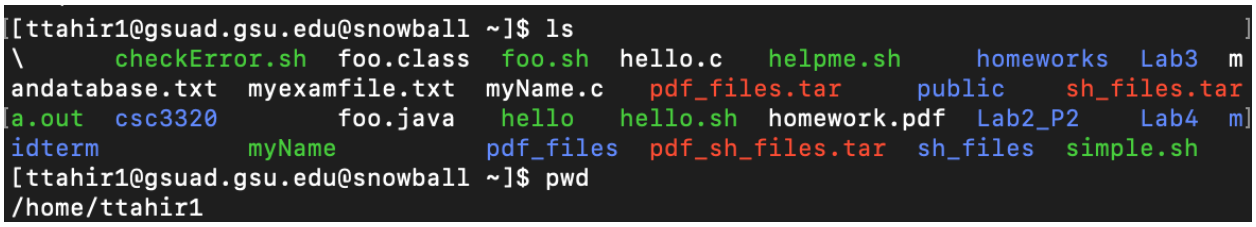
```
Phone Book Utility
-----------------
1: Display Book
2: Add Person
3: Edit Person
4: Find Person
5: Delete Person
6: Exit

Enter [1-6]
5
Who would ou like to delete?
Enter First Name:
John
John has been removed.
Phone Book Utility
-----------------
1: Display Book
2: Add Person
3: Edit Person
4: Find Person
5: Delete Person
6: Exit

Enter [1-6]
1
First Name: Bob , Last Name: Marley ,  Number: 1800789765 , Address: 210 Pidemont
Phone Book Utility
-----------------
1: Display Book
2: Add Person
3: Edit Person
4: Find Person
5: Delete Person
6: Exit

Enter [1-6]
6
Phone Book program closed
```

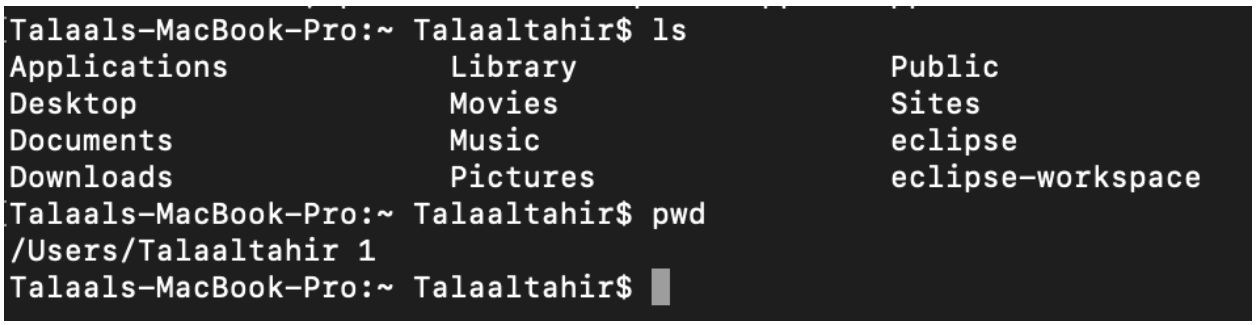
```
#!/bin/bash
# Talaal Tahir
# Ttahir1@student.gsu.edu
# phonebook utilityh

awnser="yes"
while [ $awnser == "yes" ]
do
echo "Phone Book Utility"
echo "------------------"
echo "1: Display Book"
echo "2: Add Person"
echo "3: Edit Person"
echo "4: Find Person"
echo "5: Delete Person"
echo "6: Exit"
echo " "
echo "Enter [1-6]"

read choice

if [ $choice == 1 ]
then
if [ -s phonebook.txt ]
then
echo "The phone book contains:"
sort +0 -2 phonebook.txt
else
echo "phonebook is empty"
fi

elif [ $choice == 2 ]
then
echo "Enter First,Last Name, Number, and address"
echo "First Name:"
read fname
echo "Last Name:"
read lname
echo "Number:"
read number
echo "Address:"
read address
echo "First Name: $fname , Last Name: $lname ,  Number: $number , Address: $address">>phonebook.txt
echo "New info added"
```

```
elif [ $choice == 3 ]
then
echo "Who would you like to edit?"
read who
if fgrep -q $who phonebook.txt;
then
fgrep $who phonebook.txt
echo "What would you like to edit"
read what
echo "What would you like to edit it to?"
read edit
sed -i "s/$what/$edit/" phonebook.txt
else
echo "Person not in phone book"
fi

elif [ $choice == 4 ]
then
echo "Who would you like to find?"
echo "Enter First Name:"
read ffind
echo "Enter Last Name:"
read lfind
fgrep $ffind phonebook.txt | fgrep $lfind

elif [ $choice == 5 ]
then
echo "Who would ou like to delete?"
echo "Enter First Name:"
read -r fdelete
sed -i "/$fdelete/d" phonebook.txt
echo "$fdelete has been removed."

elif [ $choice == 6 ]
then
echo "Phone Book program closed"
break
fi
done
```

Question 5

1. What is the use of a shell?
    a. It helps understand and execute commands that a user enters. It is used instead of a graphical user interface. It allows for the user to interact with the system.
2. Is there any difference between the shell that you see on your PC versus that you see on the snowball server upon login. If yes, what are they? Provide screenshots for examples.
    a. Yes, when on the PC I have complete access to all the files and folders because I am the owner and main user of the PC while on the snowball server I am limited to what the owner allows me to see. I have only access to /home/ttahir directory while, on my pc i have access to a lot more directories.





3. What are the elements in a computer (software and hardware) that enable the understanding and interpretation of a C program?
    a. Three steps are necessary before a computer can execute code, preprocessing, compiling and Linking. The preprocessor obeys commands that begin with #. The compiler translates the program into machine instructions and the linker links the code from the compiler and anything else necessary to execute the code. In linux to compile the code you can use $cc. On the hardware side most code is CPU and memory intensive.
4. The “printf()” C command is used for printing anything on the screen. In bash we use the command “echo ”. What is the difference (if any) in terms of how the computer interprets and executes these commands?
    a. The main difference between printf and echo is that printf does not automatically advance to the next output line when it is done printing. To advance to the next output line you must include \n in the string to be printed. While echo by default appends a new line, to stop this you would have to use the -n option,
5. What do these shell commands do? “ssh”, “scp” and “wget”. Describe briefly using an example that you have executed using the snowball server.
    a. ssh - Stands for Secure Shell or Secure Socket Shell, which gives users a secure way to access a system over an unsecured network.
        i. Ex: To connect to the snowball server we put ssh to create a connection

    ii. ssh Ttahir1@snowball.cs.gsu.edu

b. scp - Stands for Secure copy, which allows secure transferring of files between the local host and the remot or host or between two remote hosts
    i. Ex: To get files from your local computer to the snowball server
    ii. Scp /home/Talaal/desktop /home/Ttahir1/
c. wget - is used to download files from a server, this will let you download even when the user is not logged onto the server.
    i. Ex: To download files from the snowball server.