

CSc 3320: Systems Programming

Fall 2021

Midterm 1 Redo: 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: Aparna Mandapaka

Campus ID: amandapaka2

Panther #: 002553236

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*
 - a. I got this question wrong because the code was not running correctly, so I came up with a different approach to approach this problem. I have created a new .sh file and named it "helpme.sh" and have created a new approachable way code. In this code, I have created a "while" loop and two "elif" statements. First I set the variable "x = 1" and then I set the "counter="0"" which basically sets the conditions to make the counter know when to stop the program and the loop. Then I have used the command "echo" to display a message to the user about the command that they want to search and then executed the "read" command to read the user input. Then created a variable name "match_condition" and set it equal to "\$(grep -i ^\$searched\ (mandatabase.txt | wc -l)", where the grep command finds the name that the user inputted in the mandatabase text file, the wc is used to count the number of lines in the file parameter and -l list the files and the directories. Then I have used the "if" statement like if the match condition is matched with the one in the database text file then the while loop comes into play reads the line and then I used "do" if the line equals the or matches with the name that user inputted and then "echo "\$ln"" prints the main function of the command if it found the command in the man database textfile, then the counter = "1". My next command line was "elif [["\$counter" -eq "1" && \$ln == *\${searched^^} "(1)"]]", where the -eq acts as the binary comparison operator between the "\$counter" and "1" and then that line is set equal to the input that was given by the user, then the program prints the line by searching the word that user has entered and the counter = "0". I created the same command as above for the next line but this time if the line does not equal to the search then it prints the line and then done, if the command or word that the user entered is not found in the database text then the user is prompted that "sorry, I cannot help you". Overall when the ./helpme.sh is executed it will match between the input and input (1). If the input doesn't match then it displays the error

The 10 commands I used are

wc, sed, mkdir, cp, tar, kill, sort, echo, ssh, and sudo

To run the file type in **./helpme.sh**

```
NAME
chmod - change file mode bits

SYNOPSIS
chmod [OPTION]... MODE[,MODE]... FILE...
chmod [OPTION]... OCTAL-MODE FILE...
-bash-4.2$ ./helpme.sh
enter the command you want to be searched
awk
sorry, I cannot help you
-bash-4.2$ ./helpme.sh
enter the command you want to be searched
ls
LS(1)                                User Commands                                LS(1)

NAME
ls - list directory contents

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

PWD(1)                                User Commands                                PWD(1)

NAME
pwd - print name of current/working directory

SYNOPSIS
pwd [OPTION]...

SED(1)                                User Commands                                SED(1)

NAME
sed - stream editor for filtering and transforming text

SYNOPSIS
sed [OPTION]... {script-only-if-no-other-script} [input-file]...

ECHO(1)                                User Commands                                ECHO(1)

NAME
echo - display a line of text

SYNOPSIS
echo [SHORT-OPTION]... [STRING]...
echo LONG-OPTION
MKDIR(1)                                User Commands                                MKDIR(1)

NAME
mkdir - make directories

SYNOPSIS
mkdir [OPTION]... DIRECTORY...

GREP(1)                                General Commands Manual                                GREP(1)

NAME
grep, egrep, fgrep - print lines matching a pattern

SYNOPSIS
grep [OPTIONS] PATTERN [FILE...]
grep [OPTIONS] [-e PATTERN | -f FILE] [FILE...]
CHMOD(1)                                User Commands                                CHMOD(1)

NAME
chmod - change file mode bits

SYNOPSIS
chmod [OPTION]... MODE[,MODE]... FILE...
chmod [OPTION]... OCTAL-MODE FILE...
-bash-4.2$
```

OpenSSH SSH client

```
#!/bin/bash
x=1
counter="0"

echo "enter the command you want to be searched"
read searched \c
match_condition="$(grep -i ^$searched\ mandatabase.txt | wc -l)"
if [ $match_condition -ge $x ]
then

while read ln; do
    if [[ $ln == ${searched^^}("*"]
    then
        echo "$ln"
        counter="1"
    elif [[ "$counter" -eq "1" && $ln == *${searched^^}(1) ]]
    then
        echo "$ln"
        counter="0"
    elif [[ "$counter" -eq "1" && $ln != *${searched^^}(1) ]]
    then
        echo "$ln"
    fi
done < mandatabase.txt

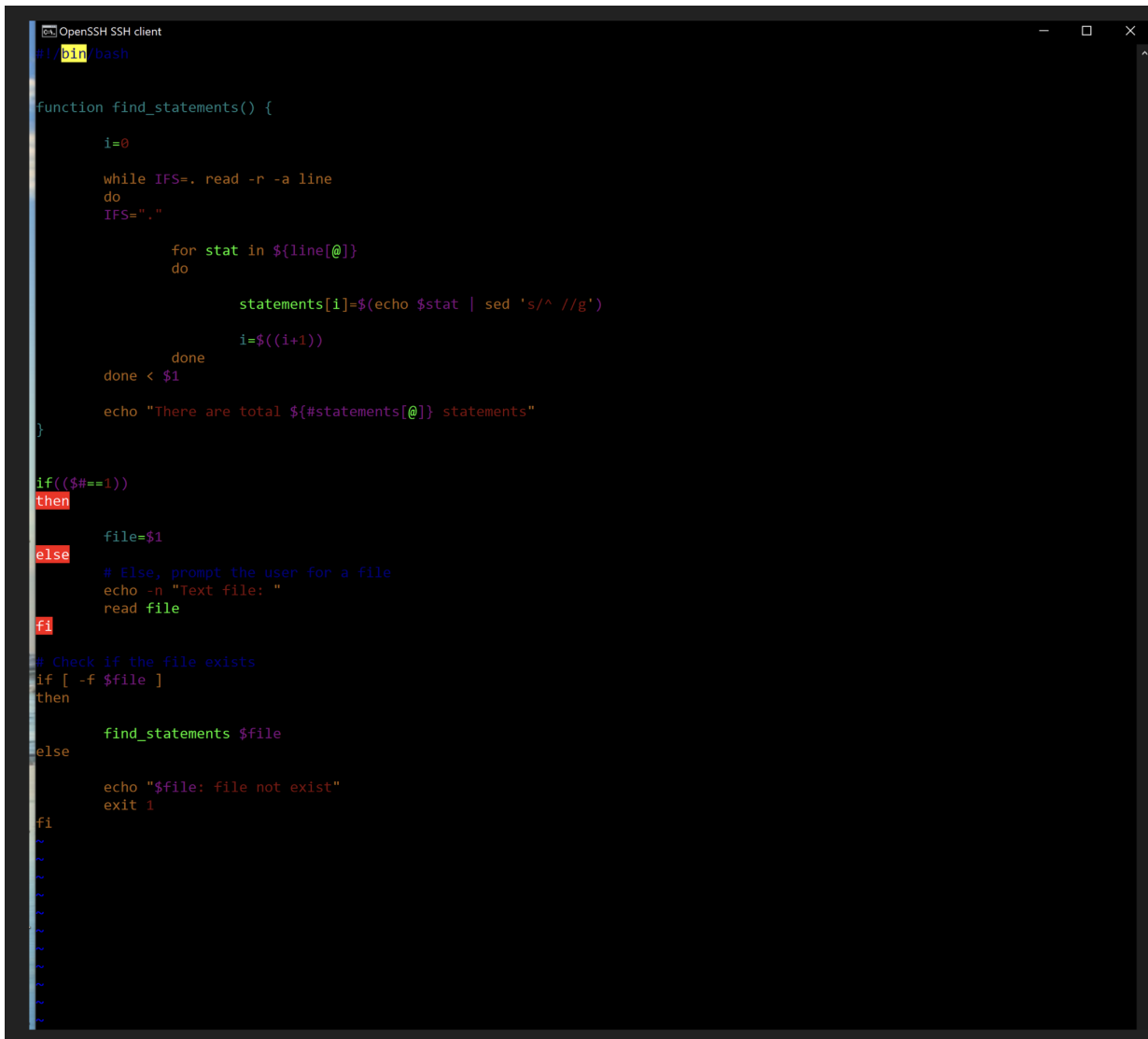
else
    echo "sorry, I cannot help you"
fi
```

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 (use any FTP tool such as Putty or Filezilla to copy the file from your computer to the remote snowball server machine: see Lab 6).
 - 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.

For the `myexamfile.txt` I have approached this question by first selecting a Wikipedia page that was not too long or too short and when I found the Wikipedia page I have copied it onto a google doc. Then I created `myexamfile.txt` and pasted the text that I found on Wikipedia over there. After that, I created a `.sh` file that contains the program to run by the Unix shell. In the script, I have created the main function that would be able to find the statements, because this was one of the questions which were to find the number of statements in the text. So in general something is considered as a statement if it's the collection of text between two periods. So to do that I have created a variable that initializes the array index and reads the statement or a line before the period and counts it as the statement. Then I created a for loop which counts the number of words and letters in the `myexamfile.txt`, during the for loop it compares the text with period and texts that do not have a period and then prints a statement displaying the number of words and letters that were present in the text. Finally, I created an if statement which would check if the argument has been passed or not, if the argument passes then it was set to `$1` and else if the argument fails it prompts the user for a file and reads the file, and then there is a command where it checks through the user-provided file name if the file is available, the command or the script "`find_statements`" is called and looks for the number of times or the number of statements that were present, if the file does not exist then the program displays an error stating that "file not exist". To conclude when the user executes the program using the `./question2.sh myexamfile.txt` the program will read the file which already contains a sample Wikipedia page, and we can use the command `grep` to print the results that were piped into the account that word is contained within the text file.

Question 2 output

Question 2 Part A



```
OpenSSH SSH client
#1, bin/bash

function find_statements() {

    i=0

    while IFS=. read -r -a line
    do
        IFS=","

        for stat in ${line[@]}
        do

            statements[i]=$(echo $stat | sed 's/^ //'')

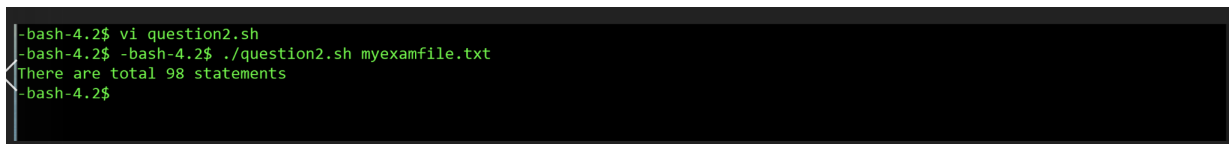
            i=$((i+1))
        done
    done < $1

    echo "There are total ${#statements[@]} statements"
}

if(($#==1))
then
    file=$1
else
    # Else, prompt the user for a file
    echo -n "Text file: "
    read file
fi

# Check if the file exists
if [ -f $file ]
then
    find_statements $file
else
    echo "$file: file not exist"
    exit 1
fi
```

Question 2 Part A output



```
-bash-4.2$ vi question2.sh
-bash-4.2$ -bash-4.2$ ./question2.sh myexamfile.txt
There are total 98 statements
-bash-4.2$
```

Question 2 part B (updated the above script)

```
function fnd_statements() {
    # initialize array index counter
    i=0

    while CLT= read -r -a line
    do
        CLT= " "

        for stat in ${line[@]}
        do
            statements[i]=${echo $stat | sed 's/^ //' }
            # increment counter
            i=$((i+1))
        done

        done < $1

        # Display the table header
        echo -e "11Words11letters"
        # initialize the array counter
        i=1

        for statement in ${statements[@]}
        do
            # Count number of words and letters
            words=$(echo $statement | wc -w)
            letters=$(echo $statement | tr -d ' ' | wc -c)
            # Display the words and letters count
            echo -e "Statement $i:\t $words\t $((letters-i))"

            i=$((i+1))
        done
    }

    # Check if argument is passed
    if [[ $# == 1 ]]
    then
        # Then, set file to $1
        file=$1
    else
        # Else, prompt the user for a file
        echo -e "Text file: "
        read file
    fi

    # Check if the file exists
    if [ -f $file ]
    then
        # then, call the function
        fnd_statements $file
    else
        # else, display error and exit
        echo "$file: file not exist"
        exit 1
    fi
}
```


OpenSSH SSH client

```
Statement 66: 6 29
Statement 67: 7 36
Statement 68: 4 24
Statement 69: 12 55
Statement 70: 13 57
Statement 71: 30 135
Statement 72: 26 131
Statement 73: 18 76
Statement 74: 11 45
Statement 75: 1 1
Statement 76: 1 1
Statement 77: 5 26
Statement 78: 12 60
Statement 79: 3 18
Statement 80: 5 28
Statement 81: 4 21
Statement 82: 2 8
Statement 83: 4 22
Statement 84: 2 22
Statement 85: 5 23
Statement 86: 5 24
Statement 87: 2 7
Statement 88: 4 22
Statement 89: 5 24
Statement 90: 6 21
Statement 91: 3 18
Statement 92: 5 25
Statement 93: 1 2
Statement 94: 1 1
Statement 95: 20 101
Statement 96: 2 13
Statement 97: 2 15
Statement 98: 4 23
-bash-4.2$
```

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.

For the calculator program, I would say this was one of the challenges and an interesting one to work with. For this program I have created a boolean and set it equal to true, it is true because whenever you check the values it compares it with the boolean. While the boolean is true, it prompts or asks the user to enter the first number and the command “read” is used, so that the system would read the users input and then prompts or asks the user for another number input and as the provide it the “read” command again reads the user input. Then gives the user the list of calculator options and then again the “read” command reads what calculator option did the user choose. The overall structure of this code was to be readable and understandable, so while approaching or doing this code, my goal was to make it look as readable as possible. In order to achieve that I have first written the commands where it mostly asks users for their input and provide them with the calculator options and then read it. I mostly used echo commands since most of the code was to do what the user wants and the echo command basically is used to print a line of text in standard output. One way that I approached this problem is through first assigning the numbers and giving out the options to the user and then creating a program that reads the number that was picked by the user, once the number is read then it gets allocated or assigned to the program and performs the wanted operations. After that, I have allocated or created two more option that allows the user to “clear and continue” and to “cancel” the whole program. For this, I have used echo again to display the clear and continue and cancel option to the user and the I have executed a command “read ch1” which is able to read the user input, and then in a separate line I have made a code for the option “clear and continue” which is if the user picks that option then the program should continue, whereas if the user picks “cancel” option then I wrote a code where the program could break and exist from the application.

Question 3

To execute the file, type in the following command

./calculator.sh

```
snowball.cs.gsu.edu - PuTTY
login as: amandapaka2
amandapaka2@snowball.cs.gsu.edu's password:
Last login: Mon Oct 11 08:12:30 2021 from c-73-7-167-165.hsd1.ga.comcast.net
+
|   GSU Computer Science
|   Instructional Server
|   SNOWBALL.cs.gsu.edu
+
-bash-4.2$ ls
ad-bk.txt      fn.txt      helpme.sh.txt  midterm      Result
address-book.txt  foo.sh      homeworks      myexamfile.txt  simple.sh
calculator      foo.sj      Lab3           output.txt    temp_course.txt
calculator.sh    hello.sh    Lab4           phonebook.sh
calculator.sj    Helpme      mandatabase    question2.sh
checkError.sh   helpme.sh   mandatabase.txt question.sh
-bash-4.2$ vi calculator.sh
-bash-4.2$ ./calculator.sh
Enter The first number:
9
Enter The second number:
5
Enter Your Choice Of Operation:
1. Addition
2. Subtraction
3. Multiplication
4. Division
5. Modulus
1
Result : 14
6) Clear and Continue
7) cancel
7
-bash-4.2$
```

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).

For the phonebook code, I have approached this problem by first creating a variable "BOOK" that would equal and store the address book text and the "BOOK" would be the name of the address book, then I created a while loop to execute the set of commands, which was basically a prompt for the user. In this program the user was given five options, "add", "list", "find", "delete" or "exist". And for each option, I have created a method and wrote a shell script for it. In the while loop, I have created a "if" statement in which I have written the shell script for each option, the first one was for the "list" option, once the program reads the user input and if they choose the option "list" then the program displays the list of stored contacts with the "Line number", "Name", "Phone Number", and "Address". In the script for the list I have coded a line `nl --number-separator=" "; "$BOOK"` where the "nl" command is used to filter the line numbers, so the shell script basically all it does is it prints the book with line numbers and is paused with less. Then I have included the elif (which is the else if statement) this was created for the "find" option, in this statement I asked the user the person they want to find, and then I executed "read" which reads the user input and then it displays the number of the format before the entries. Then I initiated the grep command to search for the name string of people that the user wants to find. Then I created another "elif" statement for the delete option, in this, the user will be asked the name that they want to delete from the phonebook and then the "read" command reads the name that the user entered and then the "sed" command is used to find and locate the name that the user wants to delete and then the file is deleted, after that the sort command is used to sort out other names after one of the names is deleted and then the cp command is used to copy the edition file into the new text file. The last option is the "add" for this I used the "elif" statement and this script is different compare to the other scripts because this interacts more with the user, by asking them the "name", "phone number" "address " and takes the input from the user and stores it into the text file. After all the inputs the user is asked if they are sure if they want to add the contact, if their response is "yes" then the name, address, and the phone number will be added to the phonebook through the command `echo "$per ; $phone ; $address" >> BOOK` where the "\$per" represents the name of the person, "\$phone" represents the phone number and "\$address" is the address and ">>" towards or pointing at BOOK appends to create a file or it creates the file (contact) that does not exist into the BOOK. But if the user chooses "no" then the data will not be appended and sends that prompt to the user saying that the contact has not been appended. After that, I have used the "fi" command which makes sure and allows me to make choice based on the success or failure of the command. After the user changes and inputs after the command "fi" I have created another

command “sort -o fn.txt ad-bk.txt” which sorts the current items in the phonebook in alphabetical order from the “Ad-bk.txt” text file. Then the command “cp fn.txt ad-bk.txt” copies the current sorted contacts into the bk.txt file. Overall, the structure of the shell script is that ./phonebook.sh is basically the main code and it makes sure to store the .txt file once an output is added or deleted. I have also included the error to show and see what the application does when an invalid statement is provided.

Question 4

To execute this type the following command

./phonebook.sh

```
OpenSSH SSH client
hl/bin/sh

BOOK="ad-bk.txt"
exit=0
while [ $exit -ne 1 ]
do
    echo "Do you want to add, list, find, delete, or exit?"
    read resp

    if [ "$resp" = "list" ]
    then
        echo "Line number:  Name ;      Phone Number;      Address ;      "
        nl --number-separator="; " $BOOK
        sort -o fn.txt ad-bk.txt
        cp fn.txt ad-bk.txt
    elif [ "$resp" = "find" ]
    then
        echo -n "What person are you trying to find?"
        read fnd
        #displays the format before the entries
        echo "          Name ;      Phone Number ;      Address  "
        grep -i $fnd $BOOK
        sort -o fn.txt ad-bk.txt
        cp fn.txt ad-bk.txt
    elif [ "$resp" = "delete" ]
    then
        echo -n "Which name should I delete: "
        read per
        sed -e "/$per/d" fn.txt | tee $BOOK
        sort -o fn.txt ad-bk.txt | cat fn.txt
        cp fn.txt ad-bk.txt
    elif [ "$resp" = "exit" ]
    then
        exit=1
    elif [ "$resp" = "add" ]
    then
        echo -n "name of person: "
        read per
        echo -n "Phone Number: "
        read phone
        echo -n "Enter the address: "
        read address
        echo "Are you sure? (y/n)"
        read res
        if [ "$res" = "y" ]
        then
            echo "$per ; $phone ; $address" >>$BOOK
        else
            echo "It has not been appended!"
        fi
        sort -o fn.txt ad-bk.txt
        cp fn.txt ad-bk.txt
    else
        echo "Error in command. Try again."
    fi
done
exit 0

~
~
~
~
~
"phonebook.sh" 59L, 1455C
```

Output

```
-bash-4.2$ ./phonebook.sh
Do you want to add, list, find, delete, or exit?
add
name of person: Kristen
Phone Number: 2028765409
Enter the address: 276 Oak Creek, Johns Creek, GA
Are you sure? (y/n)
y
Do you want to add, list, find, delete, or exit?
add
name of person: Fam
Phone Number: 9098907890
Enter the address: 456 Fram Street, Pearls, NV
Are you sure? (y/n)
n
It has not been appended!
Do you want to add, list, find, delete, or exit?
list
Line number:  Name ;      Phone Number;      Address ;
1;   Helen ; 4048790876 ; 2345 welfare crossing, Forsyth, GA
2;   Kristen ; 2028765409 ; 276 Oak Creek, Johns Creek, GA
3;   Sam ; 2340987654 ; 1324 Goldmine Street, Alpharetta, GA
Do you want to add, list, find, delete, or exit?
find
What person are you trying to find?Sam
      Name ;      Phone Number ;      Address
Sam ; 2340987654 ; 1324 Goldmine Street, Alpharetta, GA
Do you want to add, list, find, delete, or exit?
delete
Which name should I delete: Helen
Kristen ; 2028765409 ; 276 Oak Creek, Johns Creek, GA
Sam ; 2340987654 ; 1324 Goldmine Street, Alpharetta, GA
Helen ; 4048790876 ; 2345 welfare crossing, Forsyth, GA
Kristen ; 2028765409 ; 276 Oak Creek, Johns Creek, GA
Sam ; 2340987654 ; 1324 Goldmine Street, Alpharetta, GA
Do you want to add, list, find, delete, or exit?
list
Line number:  Name ;      Phone Number;      Address ;
1;   Kristen ; 2028765409 ; 276 Oak Creek, Johns Creek, GA
2;   Sam ; 2340987654 ; 1324 Goldmine Street, Alpharetta, GA
Do you want to add, list, find, delete, or exit?
exit
-bash-4.2$
```


5. Give brief answers with examples, wherever relevant

a. What is the use of a shell

- i. To approach this question I have read the textbook about the shell and then I understood that shell wide range scripting language is mostly used as an interface between the user and the UNIX operating system. To know how shell works I have created a shell file using “.sh” extension which has created a file where we can type in and enter the scripts that can be performed by the program. Through this trial, I found out that shell takes in the information or command from the program and gives the user the chance to interact with the system.

b. Is there a 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 providing a screenshot?

- i. To approach this question I have logged into the terminal on my mac book and once I logged in, I can see that most of the serves were not stopping all the commands running in the system between the data source compared to snowball serve because when I logged into the snowball server I was able to see that how it stops all the commands running in the system between the data source and also it was able to secure most of the connections within the server.

```
aparnamandapaka — -bash — 80x24
Last login: Sun Oct 10 06:29:17 on ttys000
mkdir: /Users/aparnamandapaka/.bash_sessions: No space left on device

The default interactive shell is now zsh.
To update your account to use zsh, please run `chsh -s /bin/zsh`.
For more details, please visit https://support.apple.com/kb/HT208050.
touch: /Users/aparnamandapaka/.bash_sessions/E23BA5FB-F7AD-49FA-BDBA-BB3454F66205.historynew: No space left on device
Aparnas-MacBook-Pro:~ aparnamandapaka$
```

```
[Aparnas-MacBook-Pro:~ aparnamandapaka$ Ssh amandapaka2@snowball.cs.gsu.edu ]
[amandapaka2@snowball.cs.gsu.edu's password: ]
Last login: Mon Oct 11 06:25:13 2021 from c-24-125-100-183.hsd1.ga.comcast.net
+
|   GSU Computer Science
|   Instructional Server
|   SNOWBALL.cs.gsu.edu
+
-bash-4.2$
```

- c. What are the elements in a computer (software and hardware) that enable the understanding and interpretation of a C program?
 - i. To approach this question I have I had to refer back to the C book, I learned that since C is a compiled language, it is easier for the compiler to understand C than to interpret it because C is mostly a machine-generated code and it executes the output.
- d. 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?
 - i. I have used both “printf()” and “echo” in this class and I know that these both are inbuilt commands. To see the main difference between these two printing

statements I have used echo mostly in the Midterm 1 and I have seen that echo always ends with the status zero and it usually prints the argument or statement followed by an end of the line character on the standard out. I have mostly used echo for most of my programs for Midterm 1 but I have used it a lot for the calculator problem and the phonebook problem, I used echo and wrote an argument after it which displays to the user. “printf()” gives the non-zero and it is slower than the echo. I have used printf() is C scripting in the labs and homework 2, the command printf() is mostly used in C language because it gives more control over the format and allows us to give the user the modified and specific type of output, like in the printf() command you can create a quotation mark and add whatever you want the user to know and then you can use the addition symbol to add more to that statement, like the addition sign and a method name after it would display the statement you want the user to know and any other message that you have written in another method. In printf() you can use \n for new-line and it doesn't display on the output, whereas if you do the same thing in echo it displays the “\n” in the output. Echo is the best command to use if you only want to print the scripts and message that you want your user to know, it's just like explaining everything in plain text.

- e. What do these shell commands do? “Ssh”, “scp” and “wget”. Describe briefly using an example that you have executed using the snowball server.
 - i. Again to approach this problem, I had to refer to the book and also I tried all of these commands to see how they work and process within the shell system. The command ssh which I have most frequently used to get into the snowball server, whenever I used ssh to login into the snowball server it used to or uses to make a connection between two systems, which is basically used to copy, manage and move files. Using the ssh command I can login to any computer and still have access to my files and all their stay safe and secure within the snowball server.
 - ii. The command “scp” is basically a secure copy protocol and it is used to securely transfer the computer files between the local host and remote host. I have used the scp command a lot during the class and that way I approached and tested out how the scp actually works. I have used this command to copy files on a remote server to the computer and it is also used by the ssh network protocol. I have also used this to transfer files from and to snowball to my local machine in the PowerShell.
 - iii. The command “weget” was the one that I didn't use a lot in this class since its main function was to download files from the web. But to approach this problem I have used the command wget in the terminal, to see how the wget actually works I have used it to download a file and save it with a specific

name using the command “wget -o [file_name] [URL]” and it worked I was able to install a “Terraform” and I was able to change its name.