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: Alex Siegel

Campus ID: asiegel11

Panther #: 002-41-1802

Questions 1-5 are 20pts each

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

```
[asiegel11@gsuad.gsu.edu@snowball midterm]$ ./helpme.sh
Please enter a command
cat
CAT(1)
      User Commands
                 CAT(1)

NAME
       cat - concatenate files and print on the standard output

SYNOPSIS
       cat [OPTION]... [FILE]...

DESCRIPTION
       Concatenate FILE(s), or standard input, to standard output.

       -A, --show-all
              equivalent to -vET

       -b, --number-nonblank
              number nonempty output lines, overrides -n

       -e     equivalent to -vE

       -E, --show-ends
              display $ at end of each line

       -n, --number
              number all output lines

       -s, --squeeze-blank
              suppress repeated empty output lines

       -t     equivalent to -vT

       -T, --show-tabs
              display TAB characters as ^I

       -u     (ignored)

       -v, --show-nonprinting
              use ^ and M- notation, except for LFD and TAB

       --help display this help and exit

       --version
              output version information and exit

       With no FILE, or when FILE is -, read standard input.
```

```
[asiegel11@gsuad.gsu.edu@snowball midterm]$ ./helpme.sh
Please enter a command
awk
Sorry, I cannot help you
[asiegel11@gsuad.gsu.edu@snowball midterm]$
```

```bash
#!/bin/bash

#helpme.sh
#Alex Siegel - asiegel11@student.gsu.edu

#Reads users command
echo Please enter a command
read cmd

#Lists the manual based on what command the user entered
case $cmd in

        "echo")
                sed -n '1,74 p' mandatabase.txt
        ;;

        "sed")
                sed -n '77,377 p' mandatabase.txt
        ;;

        "mv")
                sed -n '380,470 p' mandatabase.txt
        ;;

        "cp")
                sed -n '473,622 p' mandatabase.txt
        ;;

        "mkdir")
                sed -n '625,675 p' mandatabase.txt
        ;;

        "cat")
                sed -n '678,775 p' mandatabase.txt
        ;;

        "date")
                sed -n '778,966 p' mandatabase.txt
        ;;

        "rm")
                sed -n '969,1067 p' mandatabase.txt
        ;;

        "pwd")
                sed -n '1070,1113 p' mandatabase.txt
        ;;

        "df")
                sed -n '1116,1216 p' mandatabase.txt
        ;;

        *)
                echo "Sorry, I cannot help you"
        ;;
esac
#End of Program
                                                                                                                    57,1          Bot
```

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.

```
[asiegel11@gsuad.gsu.edu@snowball midterm]$ ./numbStatements.sh
There are 13 statements in myexamfile.txt
Statement | Words | Letters
1 10 56
2 23 149
3 35 224
4 17 107
5 17 100
6 20 129
7 28 188
8 12 83
9 25 141
10 41 241
11 51 312
12 13 74
13 1 1
[asiegel11@gsuad.gsu.edu@snowball midterm]$
```

```
#!/bin/bash
#numbStatements.sh
#Alex Siegel - asiegel11@student.gsu.edu

#Splits the file into statements
awk -F. '{

        print "There are " NF " statements in myexamfile.txt" #Lists
        print "Statement | Words | Letters"

        for(i = 1; i <= NF; i++) {
                n = split($i, a, " ")
                print i " " n " " length($i)

        }
}' myexamfile.txt
```

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.

```
[asiegel11@gsuad.gsu.edu@snowball midterm]$ ./calculator.sh
Please enter an expression
Valid operations are + - * / and %
Please only enter one operation
1.1+1.2
1.1 + 1.2 = 2.3
Would you like to enter another expression? (yes/no)
yes
Please enter an expression
Valid operations are + - * / and %
Please only enter one operation
6.99-1
6.99 - 1 = 5.99
Would you like to enter another expression? (yes/no)
yes
Please enter an expression
Valid operations are + - * / and %
Please only enter one operation
4*3
4 * 3 = 12
Would you like to enter another expression? (yes/no)
yes
Please enter an expression
Valid operations are + - * / and %
Please only enter one operation
12/6
12 / 6 = 2
Would you like to enter another expression? (yes/no)
yes
Please enter an expression
Valid operations are + - * / and %
Please only enter one operation
5%2
5 % 2 = 1
Would you like to enter another expression? (yes/no)
no
Thank you for using calculator.sh
[asiegel11@gsuad.gsu.edu@snowball midterm]$
```

```
#!/bin/bash
#Calculator.sh
#Alex Siegel - asiegel11

#Boolean to keep calculator running
running="yes"

# Calculator Loop
while [ "$running" = "yes" ]; do

        #Gets users input
        echo "Please enter an expression"
        echo "Valid operations are + - * / and %"
        echo "Please only enter one operation"
        read input

        # Checks which expression user inputed
        if [[ $input =~ [0-9]*\.?[0-9]*\+[0-9]*\.?[0-9]* ]]
        then
                 echo $input | awk -F+ '{print $1,"+",$2,"=",($1+$2)}'
        elif [[ $input =~ [0-9]*\.?[0-9]*-[0-9]*\.?[0-9]* ]]
        then
                 echo $input | awk -F- '{print $1,"-",$2,"=",($1-$2)}'
        elif [[ $input =~ [0-9]*\.?[0-9]*\*[0-9]*\.?[0-9]* ]]
        then
                 echo $input | awk -F* '{print $1,"*",$2,"=",($1*$2)}'
        elif [[ $input =~ [0-9]*\.?[0-9]*\/[0-9]*\.?[0-9]* ]]
        then
                 echo $input | awk -F/ '{print $1,"/",$2,"=",($1/$2)}'
        elif [[ $input =~ [0-9]*\.?[0-9]*%[0-9]*\.?[0-9]* ]]
        then
                 echo $input | awk -F% '{print $1,"%",$2,"=",($1%$2)}'
        else
                echo That is not a valid expression.
        fi

        # Asks user if they want to enter another expression
        echo "Would you like to enter another expression? (yes/no)"
        read running
done
echo "Thank you for using calculator.sh"
#End of Program
```

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).

```
Welcome to the phone book
Please choose from the following options
1: List
2: Add
3: Delete
4: Edit
5: Exit
1
Name      |      Address     |      Telephone
Would you like to continue (yes/no)
yes
Please choose from the following options
1: List
2: Add
3: Delete
4: Edit
5: Exit
2
Please enter the name of the contact
Alex Siegel
Please enter the email address of the contact
asiegel11@student.gsu.edu
Please enter the phone number of the contact using the form (xxx)xxx-xxxx
(631)704-1165
Alex Siegel has been added to the phone book
Would you like to continue (yes/no)
yes
Please choose from the following options
1: List
2: Add
3: Delete
4: Edit
5: Exit
```

```
Please choose from the following options
1: List
2: Add
3: Delete
4: Edit
5: Exit
4
Please enter a contact name
Alex s
That contact is not in the phone book
Would you like to continue (yes/no)
yes
Please choose from the following options
1: List
2: Add
3: Delete
4: Edit
5: Exit
4
Please enter a contact name
Alex Siegel
Please enter a new name for this contact
Test Student
Please enter a new email address for this contact
email1@gsu.edu
Please enter a new phone number for this contact using the form (xxx)xxx-xxxx
(111)111-1111
Would you like to continue (yes/no)
yes
Please choose from the following options
1: List
2: Add
3: Delete
4: Edit
5: Exit
1
Name      |      Address     |      Telephone
Test Student    email1@gsu.edu    (111)111-1111
Would you like to continue (yes/no)
```

```
Would you like to continue (yes/no)
yes
Please choose from the following options
1: List
2: Add
3: Delete
4: Edit
5: Exit
3
Please enter a contact name
Test Student
Test Student has been removed from the phone book
Would you like to continue (yes/no)
yes
Please choose from the following options
1: List
2: Add
3: Delete
4: Edit
5: Exit
1
Name      |      Address      |      Telephone
Would you like to continue (yes/no)
```

```bash
#!/bin/bash

#phonebook.sh
#Alex Siegel - asiegel11@student.gsu.edu


#input loop
inputLoop="yes"
echo "Welcome to the phone book"

while [ "$inputLoop" = "yes" ]; do

        #Reads input
        echo Please choose from the following options
        echo 1: List
        echo 2: Add
        echo 3: Delete
        echo 4: Edit
        echo 5: Exit
        read input

        case $input in

                #List all the things
                1 | List | list | LIST)
                echo "Name      |      Address     |      Telephone"
                sort -bf phonebook.txt
                ;;

                #Asks user for new input
                2 | add | ADD | Add)
                echo "Please enter the name of the contact"
                read name
                echo "Please enter the email address of the contact"
                read contact
                echo "Please enter the phone number of the contact using the form (xxx)xxx-xxxx"
                read number
                echo "$name    $contact    $number" >>phonebook.txt
                echo "$name has been added to the phone book"
                ;;

                #Deletes contact from phone book
                3 | delete | Delete | DELETE)
                echo "Please enter a contact name"
                read name
                if grep -q "$name" phonebook.txt; then
                        sed -i "/$name/d" phonebook.txt
                        echo "$name has been removed from the phone book"
                else
```

```
                        echo That contact is not in the phone book
                fi
                ;;

                #Asks user to enter the contact
                4 | edit | EDIT | Edit)
                echo "Please enter a contact name"
                read name
                if grep -q "$name" phonebook.txt; then
                        #Asks user to enter a new name, email and phone number for that person
                        echo "Please enter a new name for this contact"
                        read newName
                        echo "Please enter a new email address for this contact"
                        read email
                        echo "Please enter a new phone number for this contact using the form (xxx)xxx-xxxx"
                        read number
                        sed -i "s/$name.*/$newName    $email    $number/g" phonebook.txt
                else
                        echo That contact is not in the phone book
                fi
                ;;

                #Exit case
                5 | exit | EXIT | exit)
                        echo "Thank you for using the phone book"
                        exit
                ;;

                *)
                echo That is not a valid option
                ;;
        esac

        #Asks user if they want to choose another option
        echo "Would you like to continue (yes/no)"
        read inputLoop
done
echo "Thank you for using the phone book"
#End of Program
```

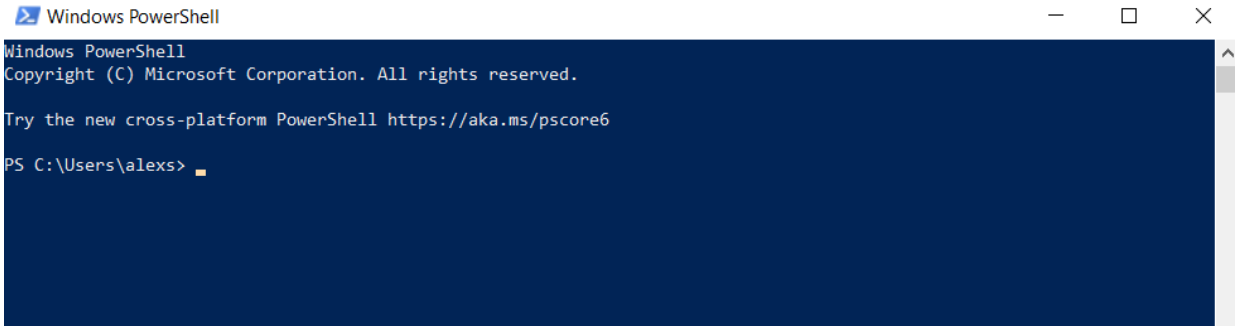
5. (4 pts each) Give brief answers with examples, wherever relevant

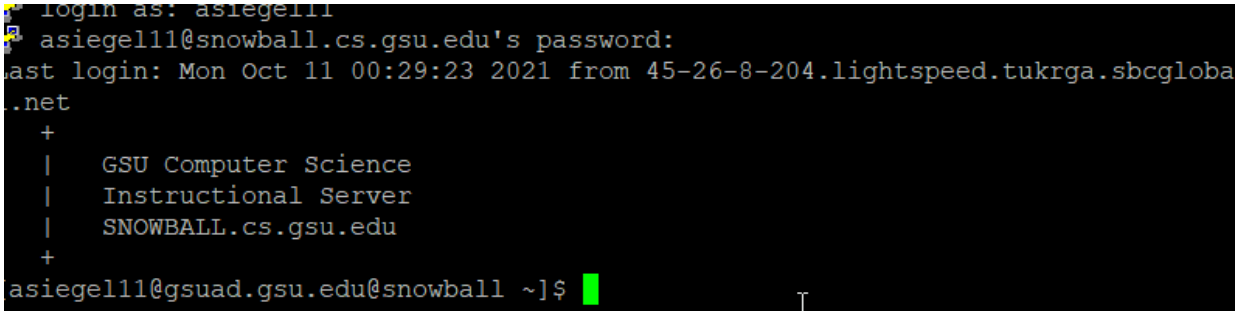
A. What is the use of a shell?

a. The shell is an interface between the user and the operating system. In UNIX, we are able to use the shell to execute commands.

B. 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. The shell on my PC has the windows copyright logo while the shell on the snowball server does not have a copyright.





C. What are the elements in a computer (software and hardware) that enable the understanding and interpretation of a C program?

   a. CPU, RAM, Hard Disk, I/O devices, Operating System, Kernel, and Network Module.

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?

   a. the “printf()” command allows for more control over the output format, while “echo” always moves to a new line after the text is printed. Furthermore, “echo” is a command in UNIX so it can be used on the command line. “printf()” is a command in C so it is on a higher “level” than “echo” is and can only be used with C interpreters.

E. What do these shell commands do? “ssh”, “scp” and “wget”. Describe briefly using an example that you have executed using the snowball server.

   a. The ssh command is used to log on to a remote machine. In class, we would do ssh snowball.cs.gsu.edu to connect to the remote server. The scp command is used to copy files onto the remote machine. wget is used to download files from the web onto a machine.