Midterm: Fall 2021

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).

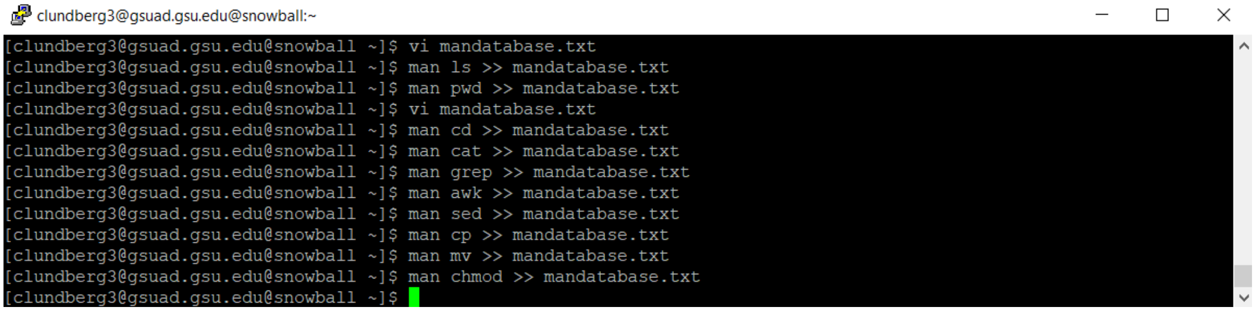
Full Name: Cassandra Lundberg

Campus ID: clundberg3

Panther #: 002345582

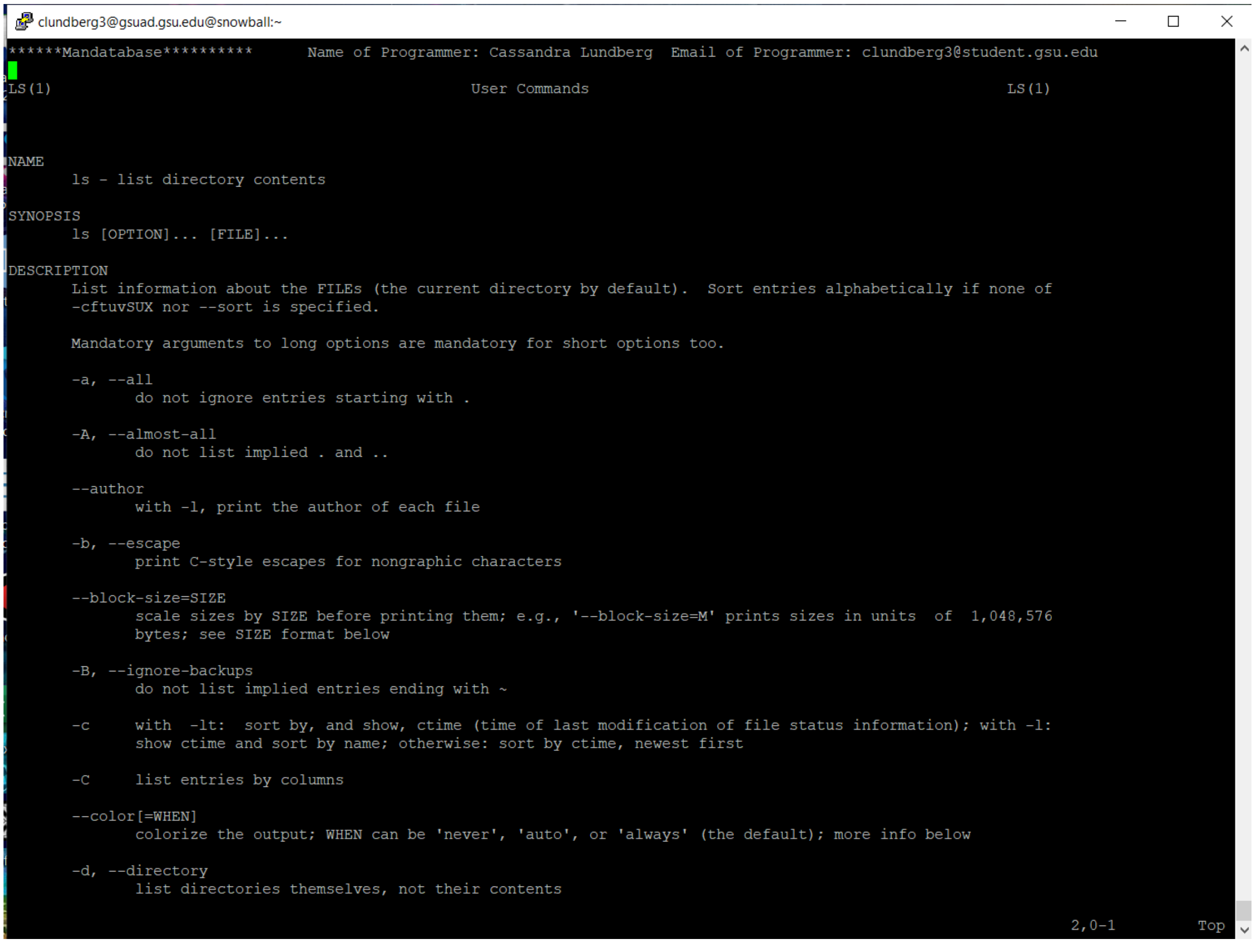
## Question: 1

Screenshot of the 10 unix commands' manuals copied onto the mandatabase.txt file:



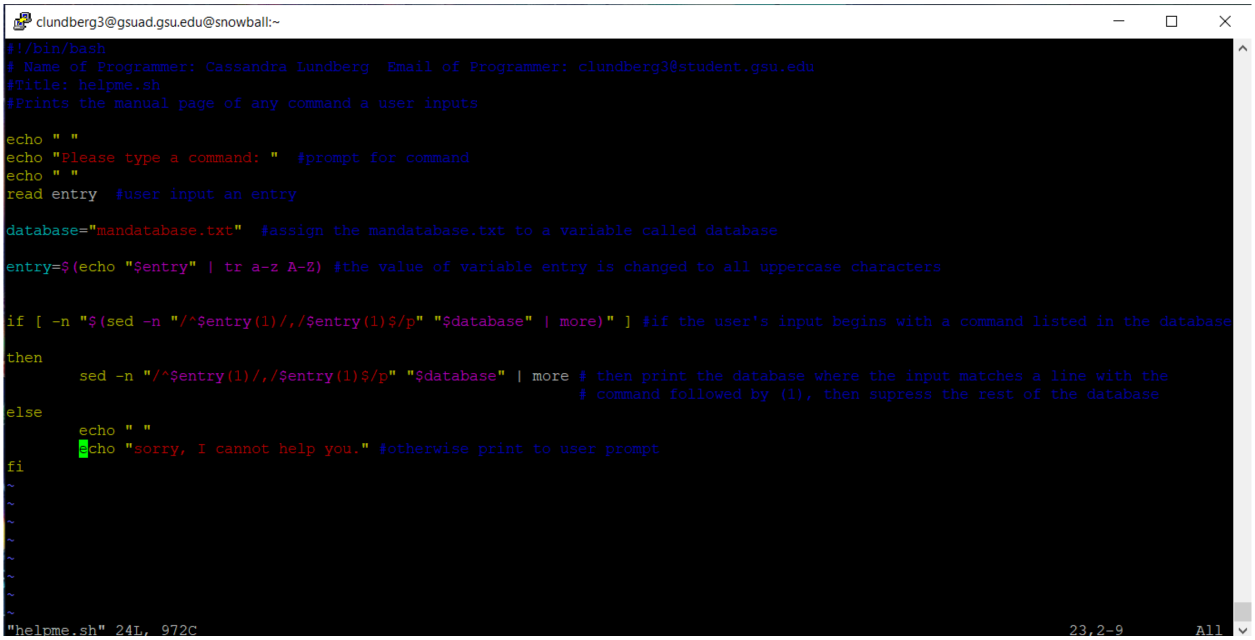
```
[clundberg3@gsuad.gsu.edu@snowball ~]$ vi mandatabase.txt
[clundberg3@gsuad.gsu.edu@snowball ~]$ man ls >> mandatabase.txt
[clundberg3@gsuad.gsu.edu@snowball ~]$ man pwd >> mandatabase.txt
[clundberg3@gsuad.gsu.edu@snowball ~]$ vi mandatabase.txt
[clundberg3@gsuad.gsu.edu@snowball ~]$ man cd >> mandatabase.txt
[clundberg3@gsuad.gsu.edu@snowball ~]$ man cat >> mandatabase.txt
[clundberg3@gsuad.gsu.edu@snowball ~]$ man grep >> mandatabase.txt
[clundberg3@gsuad.gsu.edu@snowball ~]$ man awk >> mandatabase.txt
[clundberg3@gsuad.gsu.edu@snowball ~]$ man sed >> mandatabase.txt
[clundberg3@gsuad.gsu.edu@snowball ~]$ man cp >> mandatabase.txt
[clundberg3@gsuad.gsu.edu@snowball ~]$ man mv >> mandatabase.txt
[clundberg3@gsuad.gsu.edu@snowball ~]$ man chmod >> mandatabase.txt
[clundberg3@gsuad.gsu.edu@snowball ~]$
```

Screenshot of mandatabase.txt file contents:



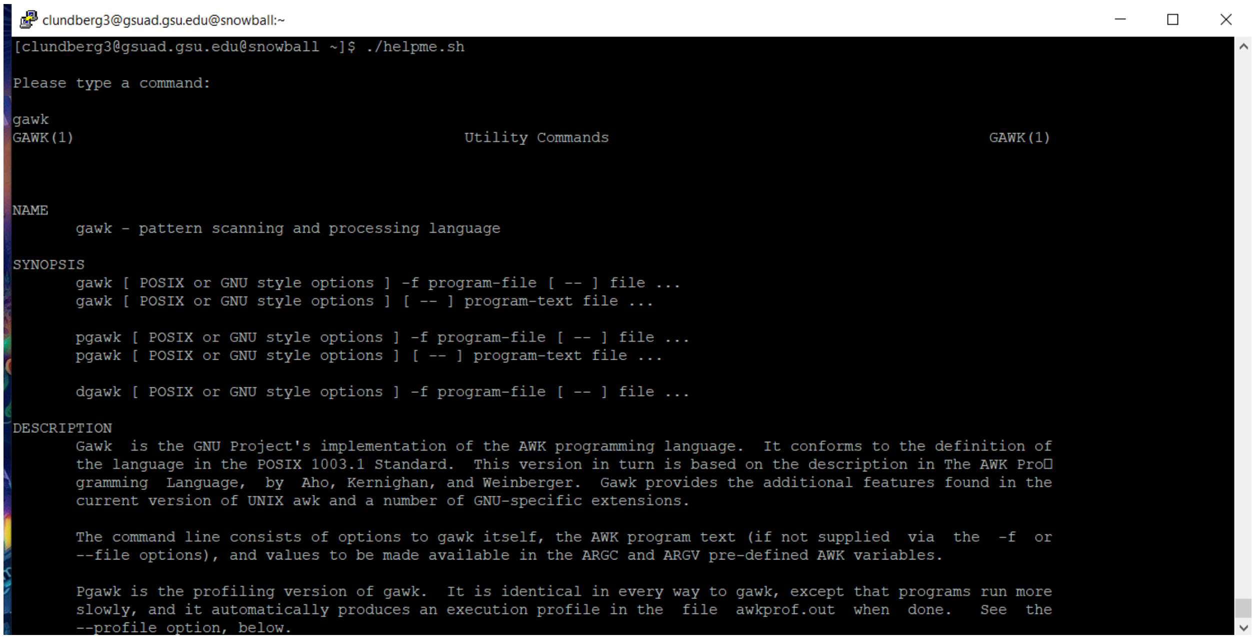
```
******Mandatabase**********     Name of Programmer: Cassandra Lundberg  Email of Programmer: clundberg3@student.gsu.edu

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

       -d, --directory
              list directories themselves, not their contents
                                                                                                   2,0-1         Top
```

Screenshot of helpme.sh shell script:



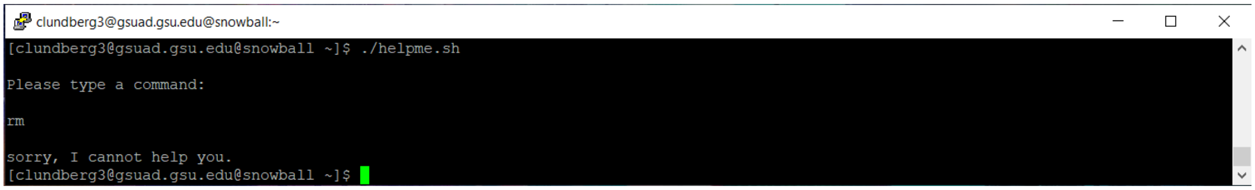
```
#!/bin/bash
# Name of Programmer: Cassandra Lundberg  Email of Programmer: clundberg3@student.gsu.edu
#Title: helpme.sh
#Prints the manual page of any command a user inputs

echo " "
echo "Please type a command: "  #prompt for command
echo " "
read entry  #user input an entry

database="mandatabase.txt"  #assign the mandatabase.txt to a variable called database

entry=$(echo "$entry" | tr a-z A-Z) #the value of variable entry is changed to all uppercase characters


if [ -n "$(sed -n "/^$entry(1)/,/$entry(1)$/p" "$database" | more)" ] #if the user's input begins with a command listed in the database

then
        sed -n "/^$entry(1)/,/$entry(1)$/p" "$database" | more # then print the database where the input matches a line with the
                                                                # command followed by (1), then supress the rest of the database
else
        echo " "
        echo "sorry, I cannot help you." #otherwise print to user prompt
fi
~
~
~
~
~
~
~
~
"helpme.sh" 24L, 972C                                                                                        23,2-9        All
```

Screenshot of shell script working when searching for the command “gawk”:



```
[clundberg3@gsuad.gsu.edu@snowball ~]$ ./helpme.sh

Please type a command:

gawk
GAWK(1)                                              Utility Commands                                              GAWK(1)



NAME
       gawk - pattern scanning and processing language

SYNOPSIS
       gawk [ POSIX or GNU style options ] -f program-file [ -- ] file ...
       gawk [ POSIX or GNU style options ] [ -- ] program-text file ...

       pgawk [ POSIX or GNU style options ] -f program-file [ -- ] file ...
       pgawk [ POSIX or GNU style options ] [ -- ] program-text file ...

       dgawk [ POSIX or GNU style options ] -f program-file [ -- ] file ...

DESCRIPTION
       Gawk  is the GNU Project's implementation of the AWK programming language.  It conforms to the definition of
       the language in the POSIX 1003.1 Standard.  This version in turn is based on the description in The AWK Pro□
       gramming  Language,  by  Aho, Kernighan, and Weinberger.  Gawk provides the additional features found in the
       current version of UNIX awk and a number of GNU-specific extensions.

       The command line consists of options to gawk itself, the AWK program text (if not supplied  via  the  -f  or
       --file options), and values to be made available in the ARGC and ARGV pre-defined AWK variables.

       Pgawk is the profiling version of gawk.  It is identical in every way to gawk, except that programs run more
       slowly, and it automatically produces an execution profile in the  file  awkprof.out  when  done.   See  the
       --profile option, below.
```

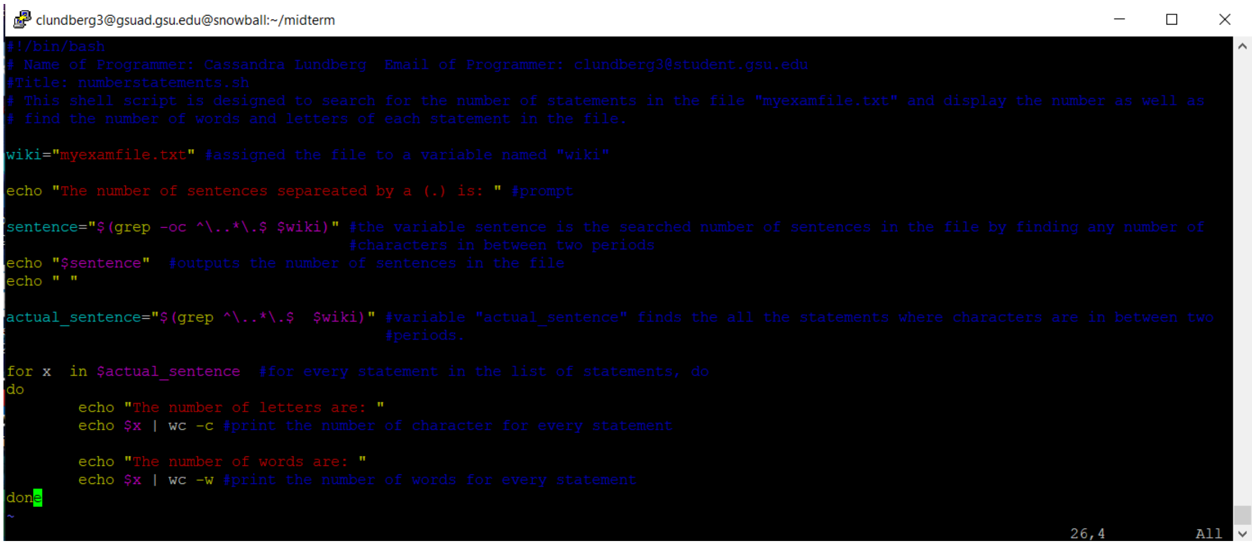
Screenshot of shell script working when searching for a command not listed in the mandatabase.txt file such as “rm”:



```
[clundberg3@gsuad.gsu.edu@snowball ~]$ ./helpme.sh

Please type a command:

rm

sorry, I cannot help you.
[clundberg3@gsuad.gsu.edu@snowball ~]$ 
```

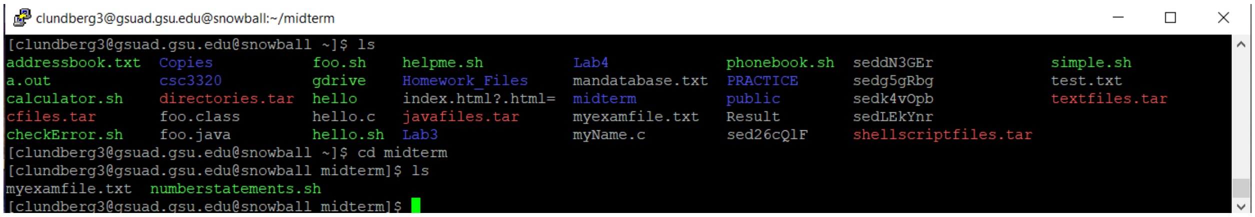
## Question: 2

Screenshot of the shell script numberstatements.sh which finds the number of statements in a file and the number of words and characters for every statement.



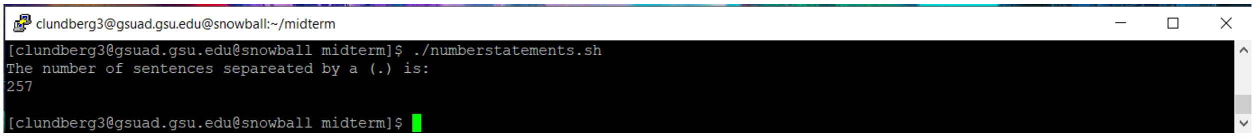
```
#!/bin/bash
# Name of Programmer: Cassandra Lundberg  Email of Programmer: clundberg3@student.gsu.edu
#Title: numberstatements.sh
# This shell script is designed to search for the number of statements in the file "myexamfile.txt" and display the number as well as
# find the number of words and letters of each statement in the file.

wiki="myexamfile.txt" #assigned the file to a variable named "wiki"

echo "The number of sentences separeated by a (.) is: " #prompt

sentence="$(grep -oc ^\..*\.$ $wiki)" #the variable sentence is the searched number of sentences in the file by finding any number of
                                      #characters in between two periods
echo "$sentence"  #outputs the number of sentences in the file
echo " "

actual_sentence="$(grep ^\..*\.$  $wiki)" #variable "actual_sentence" finds the all the statements where characters are in between two
                                          #periods.

for x  in $actual_sentence  #for every statement in the list of statements, do
do
        echo "The number of letters are: "
        echo $x | wc -c #print the number of character for every statement

        echo "The number of words are: "
        echo $x | wc -w #print the number of words for every statement
done
~
                                                                                                                   26,4          All
```

Screenshot shows the new directory called “midterm” in the home directory. The midterm directory has the file called “myexamfile” which contains the wikipedia copied text and the “numberstatements.sh” shell script which contains the code to search the “myexamfile”.



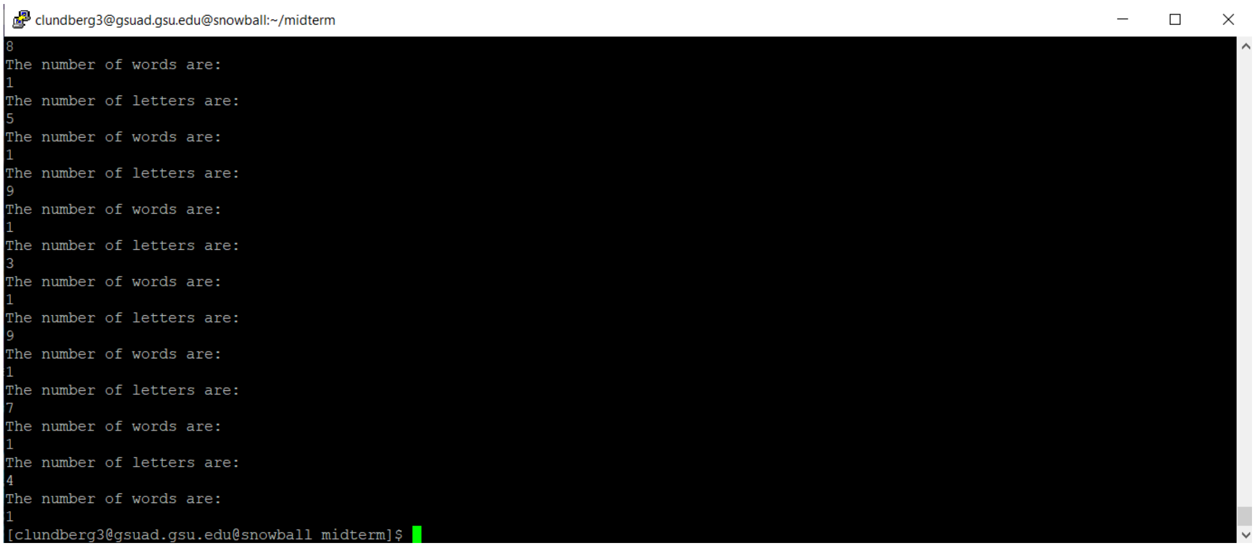
```
[clundberg3@gsuad.gsu.edu@snowball ~]$ ls
addressbook.txt  Copies           foo.sh    helpme.sh         Lab4              phonebook.sh  seddN3GEr            simple.sh
a.out            csc3320          gdrive    Homework_Files    mandatabase.txt   PRACTICE      sedg5gRbg            test.txt
calculator.sh    directories.tar  hello     index.html?.html= midterm           public        sedk4vOpb            textfiles.tar
cfiles.tar       foo.class        hello.c   javafiles.tar     myexamfile.txt    Result        sedLEkYnr
checkError.sh    foo.java         hello.sh  Lab3              myName.c          sed26cQ1F     shellscriptfiles.tar
[clundberg3@gsuad.gsu.edu@snowball ~]$ cd midterm
[clundberg3@gsuad.gsu.edu@snowball midterm]$ ls
myexamfile.txt  numberstatements.sh
[clundberg3@gsuad.gsu.edu@snowball midterm]$
```

Screenshot shows the result of part a of the question where the number of statements in the file is output.



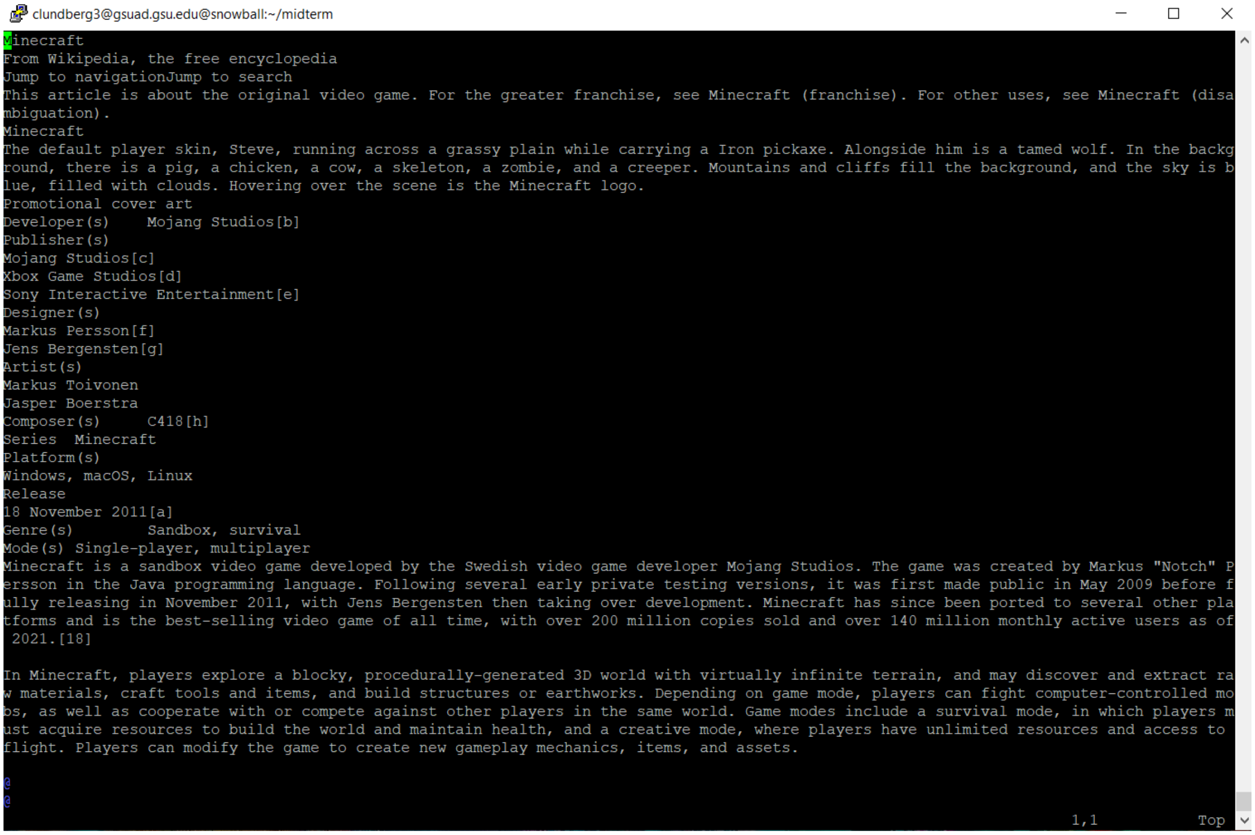
```
[clundberg3@gsuad.gsu.edu@snowball midterm]$ ./numberstatements.sh
The number of sentences separeated by a (.) is:
257

[clundberg3@gsuad.gsu.edu@snowball midterm]$
```

Screenshot shows the result of part b of the question which shows the number of words and letters in each statement.



```
8
The number of words are:
1
The number of letters are:
5
The number of words are:
1
The number of letters are:
9
The number of words are:
1
The number of letters are:
3
The number of words are:
1
The number of letters are:
9
The number of words are:
1
The number of letters are:
7
The number of words are:
1
The number of letters are:
4
The number of words are:
1
[clundberg3@gsuad.gsu.edu@snowball midterm]$
```

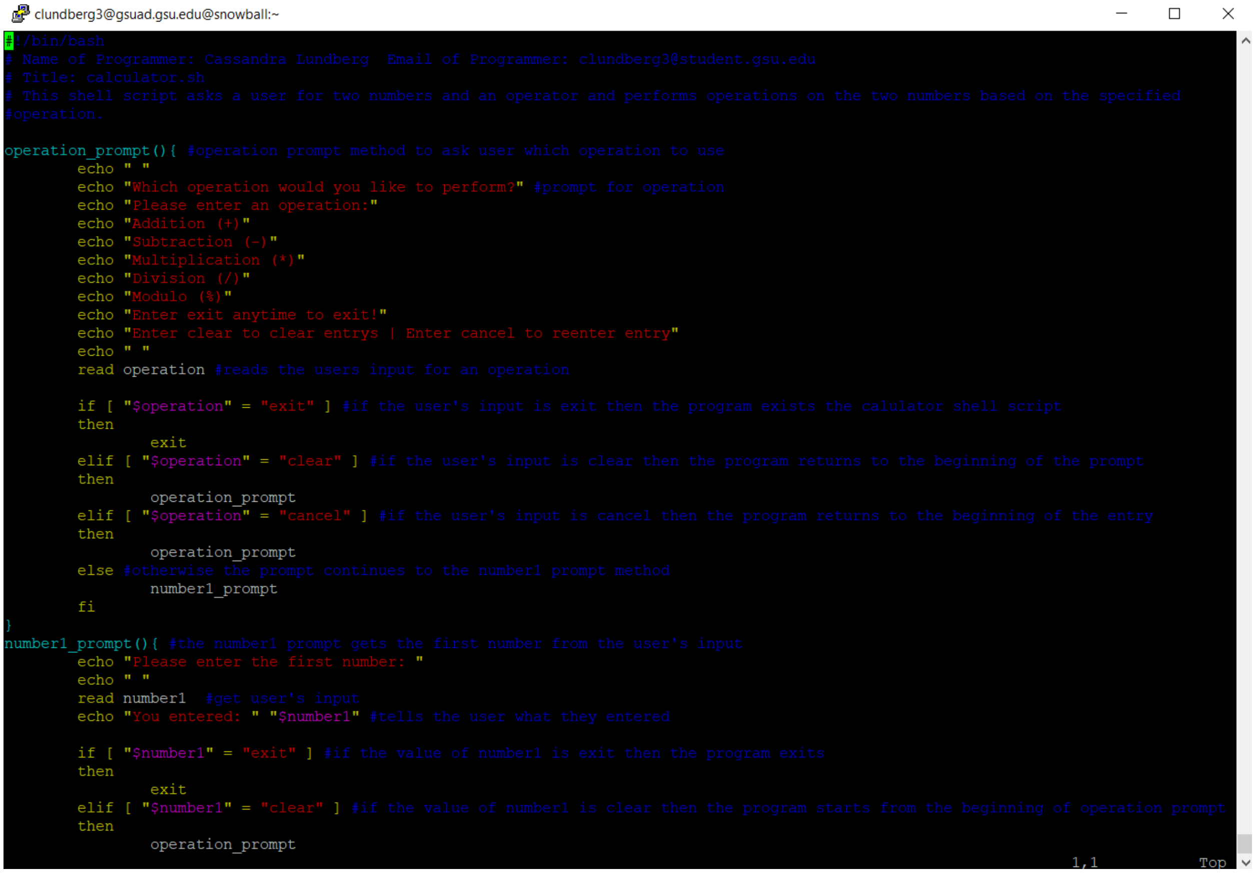
Screenshot shows “myexamfile.txt”



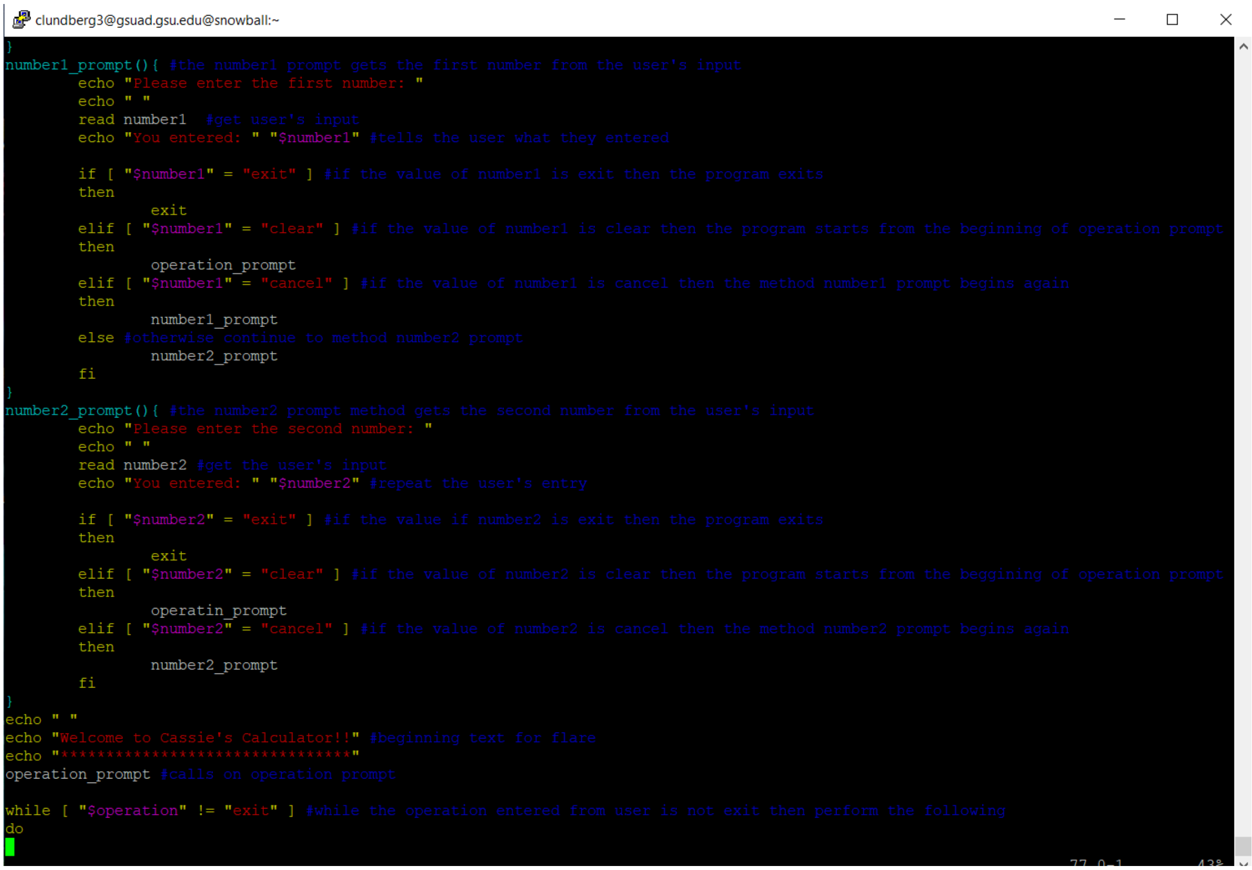
```
Minecraft
From Wikipedia, the free encyclopedia
Jump to navigationJump to search
This article is about the original video game. For the greater franchise, see Minecraft (franchise). For other uses, see Minecraft (disa
mbiguation).
Minecraft
The default player skin, Steve, running across a grassy plain while carrying a Iron pickaxe. Alongside him is a tamed wolf. In the backg
round, there is a pig, a chicken, a cow, a skeleton, a zombie, and a creeper. Mountains and cliffs fill the background, and the sky is b
lue, filled with clouds. Hovering over the scene is the Minecraft logo.
Promotional cover art
Developer(s)    Mojang Studios[b]
Publisher(s)
Mojang Studios[c]
Xbox Game Studios[d]
Sony Interactive Entertainment[e]
Designer(s)
Markus Persson[f]
Jens Bergensten[g]
Artist(s)
Markus Toivonen
Jasper Boerstra
Composer(s)     C418[h]
Series  Minecraft
Platform(s)
Windows, macOS, Linux
Release
18 November 2011[a]
Genre(s)        Sandbox, survival
Mode(s) Single-player, multiplayer
Minecraft is a sandbox video game developed by the Swedish video game developer Mojang Studios. The game was created by Markus "Notch" P
ersson in the Java programming language. Following several early private testing versions, it was first made public in May 2009 before f
ully releasing in November 2011, with Jens Bergensten then taking over development. Minecraft has since been ported to several other pla
tforms and is the best-selling video game of all time, with over 200 million copies sold and over 140 million monthly active users as of
 2021.[18]

In Minecraft, players explore a blocky, procedurally-generated 3D world with virtually infinite terrain, and may discover and extract ra
w materials, craft tools and items, and build structures or earthworks. Depending on game mode, players can fight computer-controlled mo
bs, as well as cooperate with or compete against other players in the same world. Game modes include a survival mode, in which players m
ust acquire resources to build the world and maintain health, and a creative mode, where players have unlimited resources and access to
flight. Players can modify the game to create new gameplay mechanics, items, and assets.

@
@
                                                                                                                   1,1           Top
```

## Question: 3

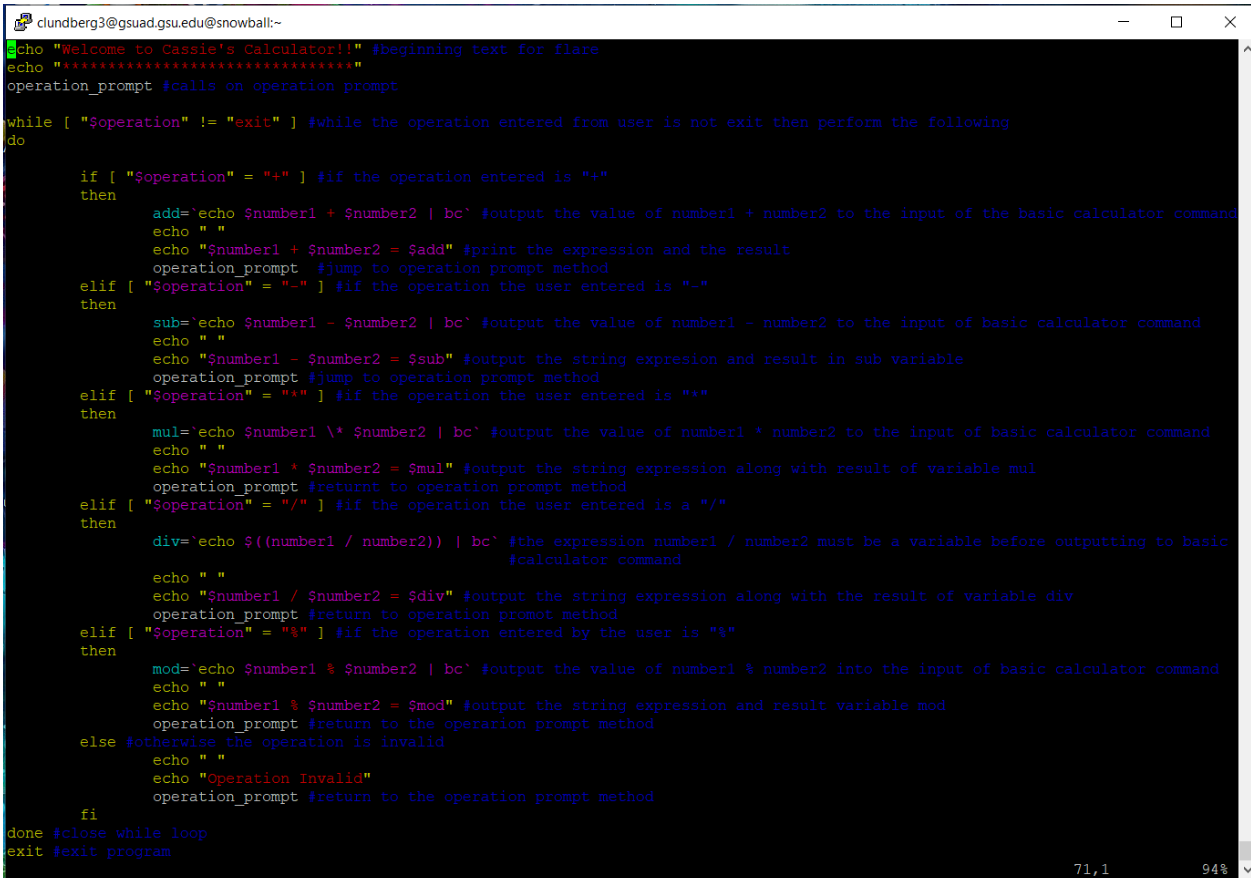
Screenshots of code of calculator.sh



```
#!/bin/bash
# Name of Programmer: Cassandra Lundberg  Email of Programmer: clundberg3@student.gsu.edu
# Title: calculator.sh
# This shell script asks a user for two numbers and an operator and performs operations on the two numbers based on the specified
#operation.

operation_prompt(){ #operation prompt method to ask user which operation to use
        echo " "
        echo "Which operation would you like to perform?" #prompt for operation
        echo "Please enter an operation:"
        echo "Addition (+)"
        echo "Subtraction (-)"
        echo "Multiplication (*)"
        echo "Division (/)"
        echo "Modulo (%)"
        echo "Enter exit anytime to exit!"
        echo "Enter clear to clear entrys | Enter cancel to reenter entry"
        echo " "
        read operation #reads the users input for an operation

        if [ "$operation" = "exit" ] #if the user's input is exit then the program exists the calulator shell script
        then
                exit
        elif [ "$operation" = "clear" ] #if the user's input is clear then the program returns to the beginning of the prompt
        then
                operation_prompt
        elif [ "$operation" = "cancel" ] #if the user's input is cancel then the program returns to the beginning of the entry
        then
                operation_prompt
        else #otherwise the prompt continues to the number1 prompt method
                number1_prompt
        fi
}
number1_prompt(){ #the number1 prompt gets the first number from the user's input
        echo "Please enter the first number: "
        echo " "
        read number1  #get user's input
        echo "You entered: " "$number1" #tells the user what they entered

        if [ "$number1" = "exit" ] #if the value of number1 is exit then the program exits
        then
                exit
        elif [ "$number1" = "clear" ] #if the value of number1 is clear then the program starts from the beginning of operation prompt
        then
                operation_prompt
                                                                                                                   1,1           Top
```



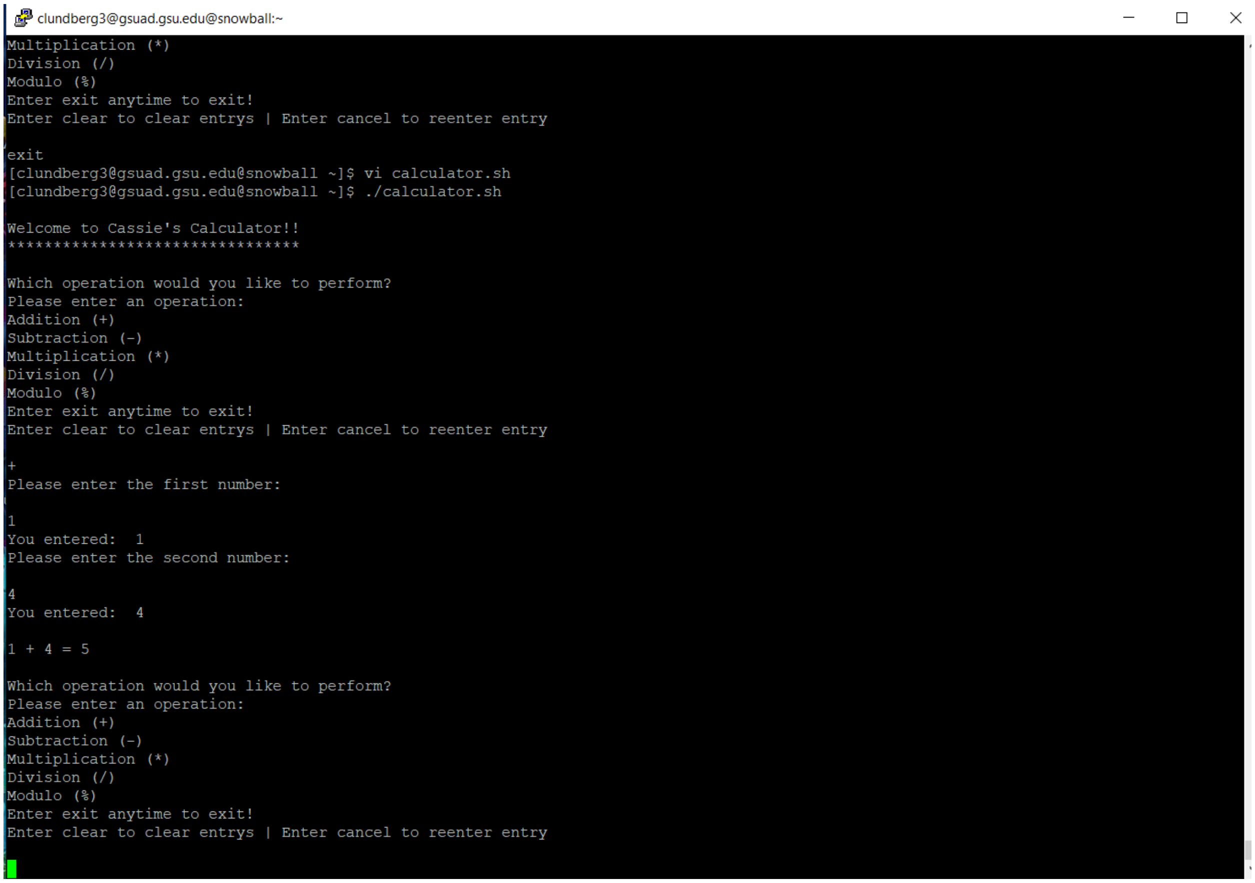
```
}
number1_prompt(){ #the number1 prompt gets the first number from the user's input
        echo "Please enter the first number: "
        echo " "
        read number1  #get user's input
        echo "You entered: " "$number1" #tells the user what they entered

        if [ "$number1" = "exit" ] #if the value of number1 is exit then the program exits
        then
                exit
        elif [ "$number1" = "clear" ] #if the value of number1 is clear then the program starts from the beginning of operation prompt
        then
                operation_prompt
        elif [ "$number1" = "cancel" ] #if the value of number1 is cancel then the method number1 prompt begins again
        then
                number1_prompt
        else #otherwise continue to method number2 prompt
                number2_prompt
        fi
}
number2_prompt(){ #the number2 prompt method gets the second number from the user's input
        echo "Please enter the second number: "
        echo " "
        read number2 #get the user's input
        echo "You entered: " "$number2" #repeat the user's entry

        if [ "$number2" = "exit" ] #if the value if number2 is exit then the program exits
        then
                exit
        elif [ "$number2" = "clear" ] #if the value of number2 is clear then the program starts from the beggining of operation prompt
        then
                operatin_prompt
        elif [ "$number2" = "cancel" ] #if the value of number2 is cancel then the method number2 prompt begins again
        then
                number2_prompt
        fi
}
echo " "
echo "Welcome to Cassie's Calculator!!" #beginning text for flare
echo "********************************"
operation_prompt #calls on operation prompt

while [ "$operation" != "exit" ] #while the operation entered from user is not exit then perform the following
do

                                                                                                          77,0-1        43%
```



```
echo "Welcome to Cassie's Calculator!!" #beginning text for flare
echo "********************************"
operation_prompt #calls on operation prompt

while [ "$operation" != "exit" ] #while the operation entered from user is not exit then perform the following
do

        if [ "$operation" = "+" ] #if the operation entered is "+"
        then
                add=`echo $number1 + $number2 | bc` #output the value of number1 + number2 to the input of the basic calculator command
                echo " "
                echo "$number1 + $number2 = $add" #print the expression and the result
                operation_prompt  #jump to operation prompt method
        elif [ "$operation" = "-" ] #if the operation the user entered is "-"
        then
                sub=`echo $number1 - $number2 | bc` #output the value of number1 - number2 to the input of basic calculator command
                echo " "
                echo "$number1 - $number2 = $sub" #output the string expresion and result in sub variable
                operation_prompt #jump to operation prompt method
        elif [ "$operation" = "*" ] #if the operation the user entered is "*"
        then
                mul=`echo $number1 \* $number2 | bc` #output the value of number1 * number2 to the input of basic calculator command
                echo " "
                echo "$number1 * $number2 = $mul" #output the string expression along with result of variable mul
                operation_prompt #returnt to operation prompt method
        elif [ "$operation" = "/" ] #if the operation the user entered is a "/"
        then
                div=`echo $((number1 / number2)) | bc` #the expression number1 / number2 must be a variable before outputting to basic
                                                       #calculator command
                echo " "
                echo "$number1 / $number2 = $div" #output the string expression along with the result of variable div
                operation_prompt #return to operation promot method
        elif [ "$operation" = "%" ] #if the operation entered by the user is "%"
        then
                mod=`echo $number1 % $number2 | bc` #output the value of number1 % number2 into the input of basic calculator command
                echo " "
                echo "$number1 % $number2 = $mod" #output the string expression and result variable mod
                operation_prompt #return to the operarion prompt method
        else #otherwise the operation is invalid
                echo " "
                echo "Operation Invalid"
                operation_prompt #return to the operation prompt method
        fi
done #close while loop
exit #exit program
                                                                                                                   71,1          94%
```

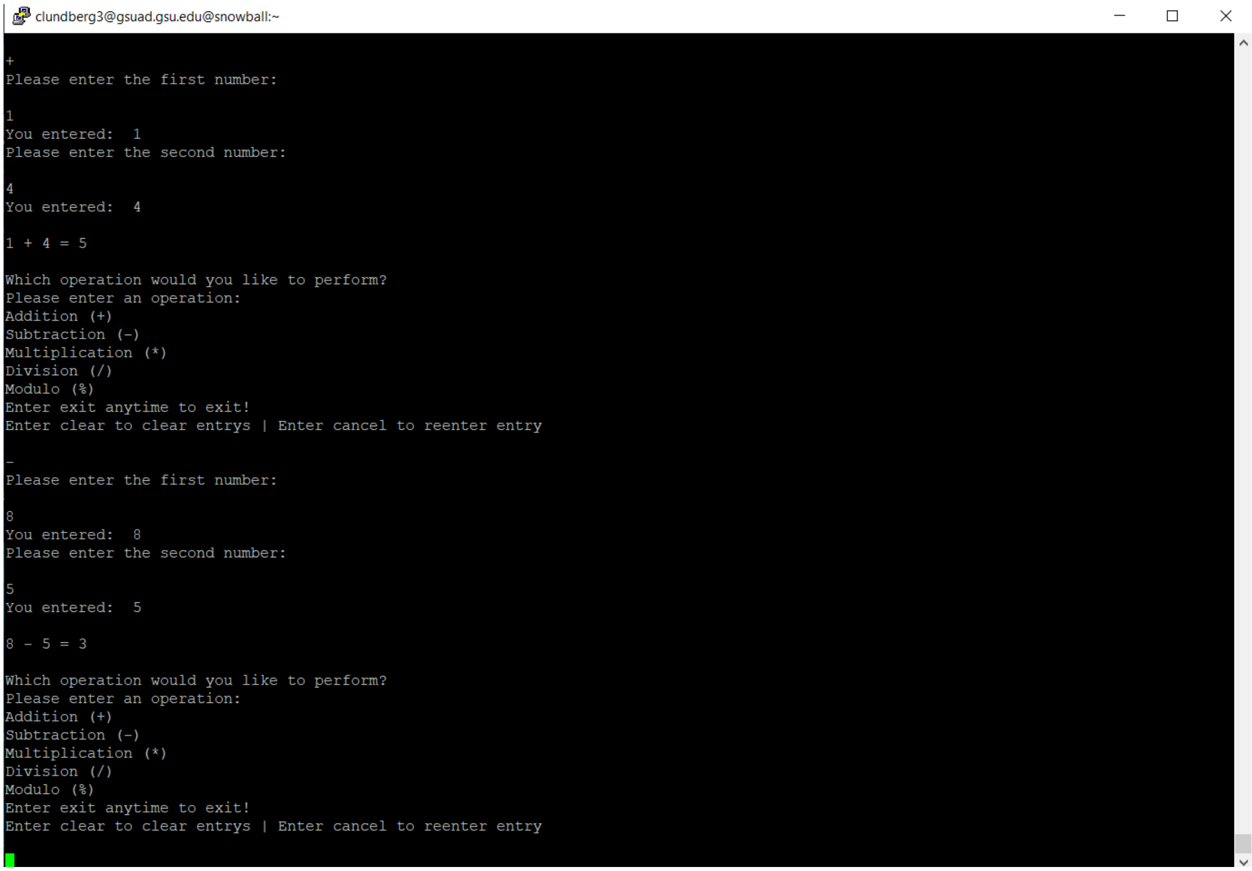
Screenshots of operations:

## Addition:



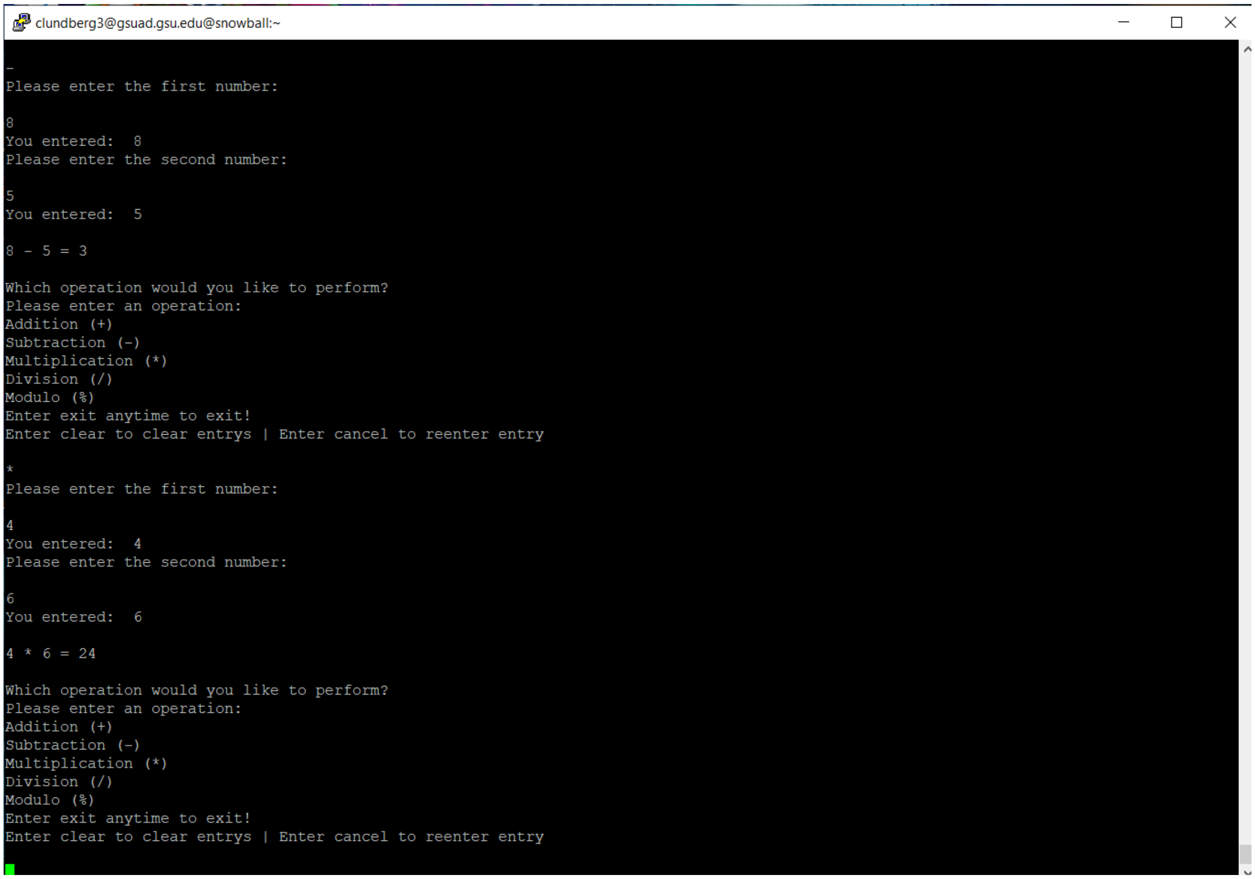
```
Multiplication (*)
Division (/)
Modulo (%)
Enter exit anytime to exit!
Enter clear to clear entrys | Enter cancel to reenter entry

exit
[clundberg3@gsuad.gsu.edu@snowball ~]$ vi calculator.sh
[clundberg3@gsuad.gsu.edu@snowball ~]$ ./calculator.sh

Welcome to Cassie's Calculator!!
********************************

Which operation would you like to perform?
Please enter an operation:
Addition (+)
Subtraction (-)
Multiplication (*)
Division (/)
Modulo (%)
Enter exit anytime to exit!
Enter clear to clear entrys | Enter cancel to reenter entry

+
Please enter the first number:

1
You entered:  1
Please enter the second number:

4
You entered:  4

1 + 4 = 5

Which operation would you like to perform?
Please enter an operation:
Addition (+)
Subtraction (-)
Multiplication (*)
Division (/)
Modulo (%)
Enter exit anytime to exit!
Enter clear to clear entrys | Enter cancel to reenter entry
```

Subtraction:



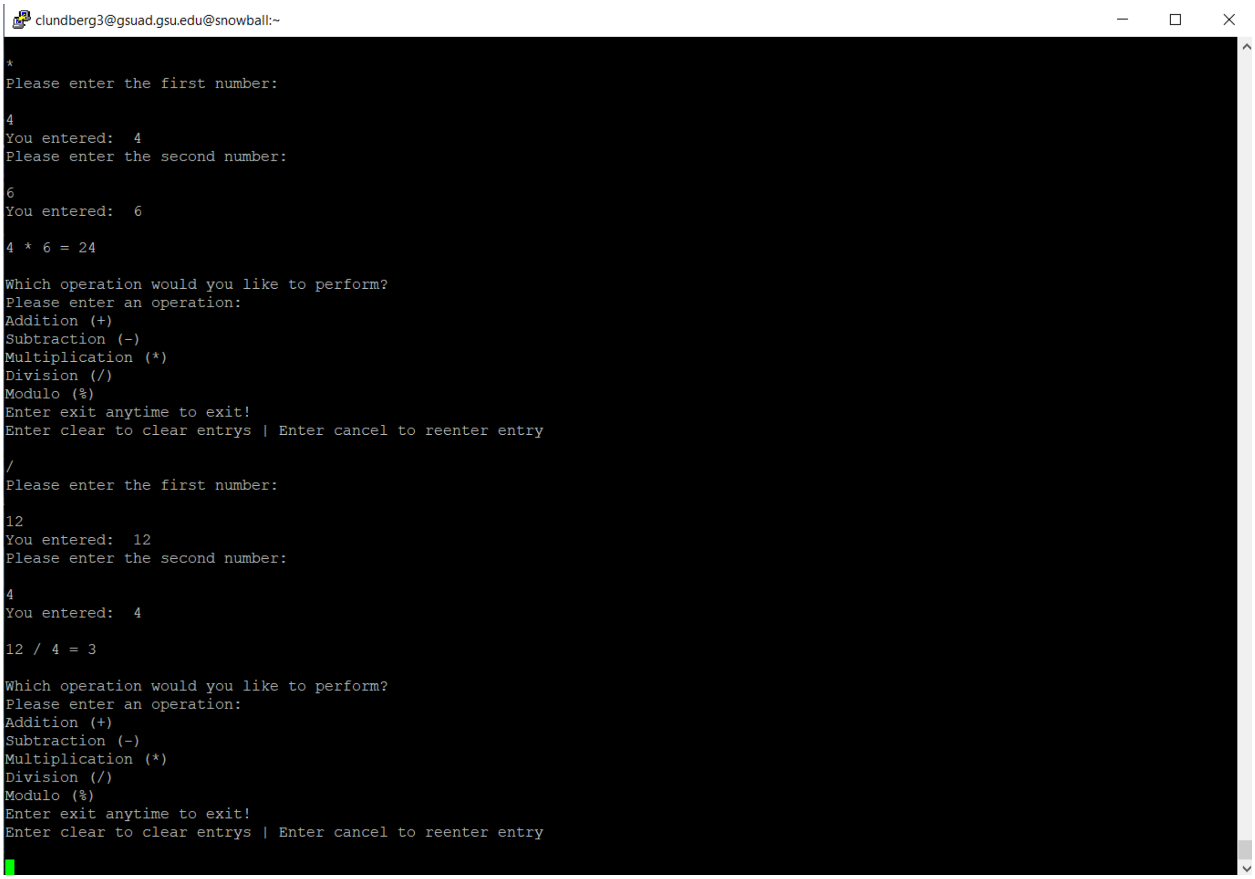
```
+
Please enter the first number:

1
You entered:  1
Please enter the second number:

4
You entered:  4

1 + 4 = 5

Which operation would you like to perform?
Please enter an operation:
Addition (+)
Subtraction (-)
Multiplication (*)
Division (/)
Modulo (%)
Enter exit anytime to exit!
Enter clear to clear entrys | Enter cancel to reenter entry

-
Please enter the first number:

8
You entered:  8
Please enter the second number:

5
You entered:  5

8 - 5 = 3

Which operation would you like to perform?
Please enter an operation:
Addition (+)
Subtraction (-)
Multiplication (*)
Division (/)
Modulo (%)
Enter exit anytime to exit!
Enter clear to clear entrys | Enter cancel to reenter entry

```

Multiplication:



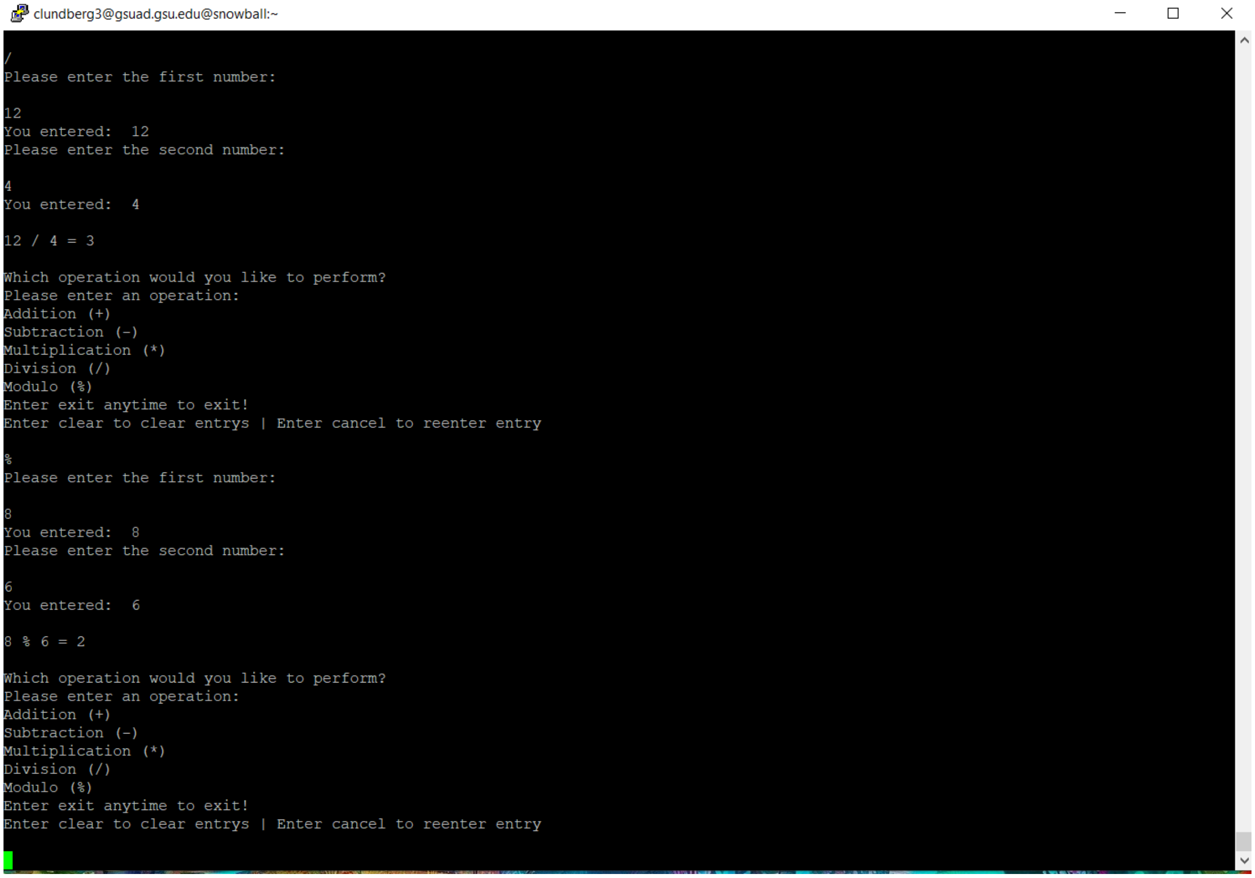
```
-
Please enter the first number:

8
You entered:  8
Please enter the second number:

5
You entered:  5

8 - 5 = 3

Which operation would you like to perform?
Please enter an operation:
Addition (+)
Subtraction (-)
Multiplication (*)
Division (/)
Modulo (%)
Enter exit anytime to exit!
Enter clear to clear entrys | Enter cancel to reenter entry

*
Please enter the first number:

4
You entered:  4
Please enter the second number:

6
You entered:  6

4 * 6 = 24

Which operation would you like to perform?
Please enter an operation:
Addition (+)
Subtraction (-)
Multiplication (*)
Division (/)
Modulo (%)
Enter exit anytime to exit!
Enter clear to clear entrys | Enter cancel to reenter entry
```

## Division:



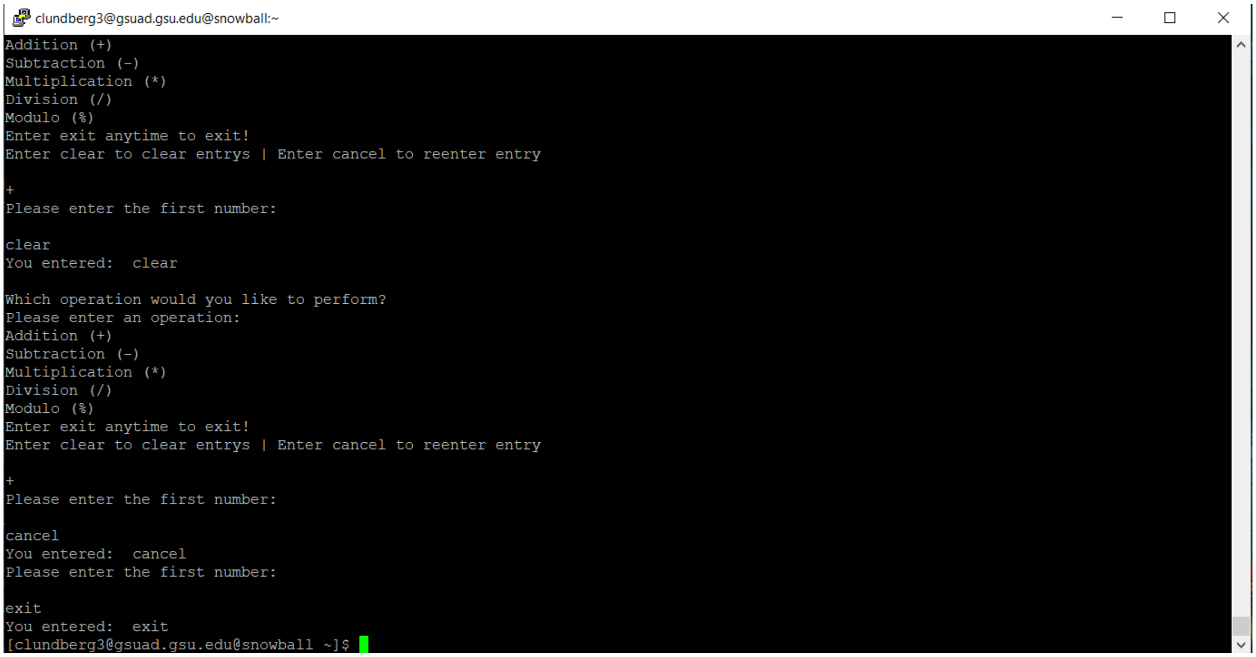
```
*
Please enter the first number:

4
You entered:  4
Please enter the second number:

6
You entered:  6

4 * 6 = 24

Which operation would you like to perform?
Please enter an operation:
Addition (+)
Subtraction (-)
Multiplication (*)
Division (/)
Modulo (%)
Enter exit anytime to exit!
Enter clear to clear entrys | Enter cancel to reenter entry

/
Please enter the first number:

12
You entered:  12
Please enter the second number:

4
You entered:  4

12 / 4 = 3

Which operation would you like to perform?
Please enter an operation:
Addition (+)
Subtraction (-)
Multiplication (*)
Division (/)
Modulo (%)
Enter exit anytime to exit!
Enter clear to clear entrys | Enter cancel to reenter entry
```

## Modulo:



```
/
Please enter the first number:

12
You entered:  12
Please enter the second number:

4
You entered:  4

12 / 4 = 3

Which operation would you like to perform?
Please enter an operation:
Addition (+)
Subtraction (-)
Multiplication (*)
Division (/)
Modulo (%)
Enter exit anytime to exit!
Enter clear to clear entrys | Enter cancel to reenter entry

%
Please enter the first number:

8
You entered:  8
Please enter the second number:

6
You entered:  6

8 % 6 = 2

Which operation would you like to perform?
Please enter an operation:
Addition (+)
Subtraction (-)
Multiplication (*)
Division (/)
Modulo (%)
Enter exit anytime to exit!
Enter clear to clear entrys | Enter cancel to reenter entry
```

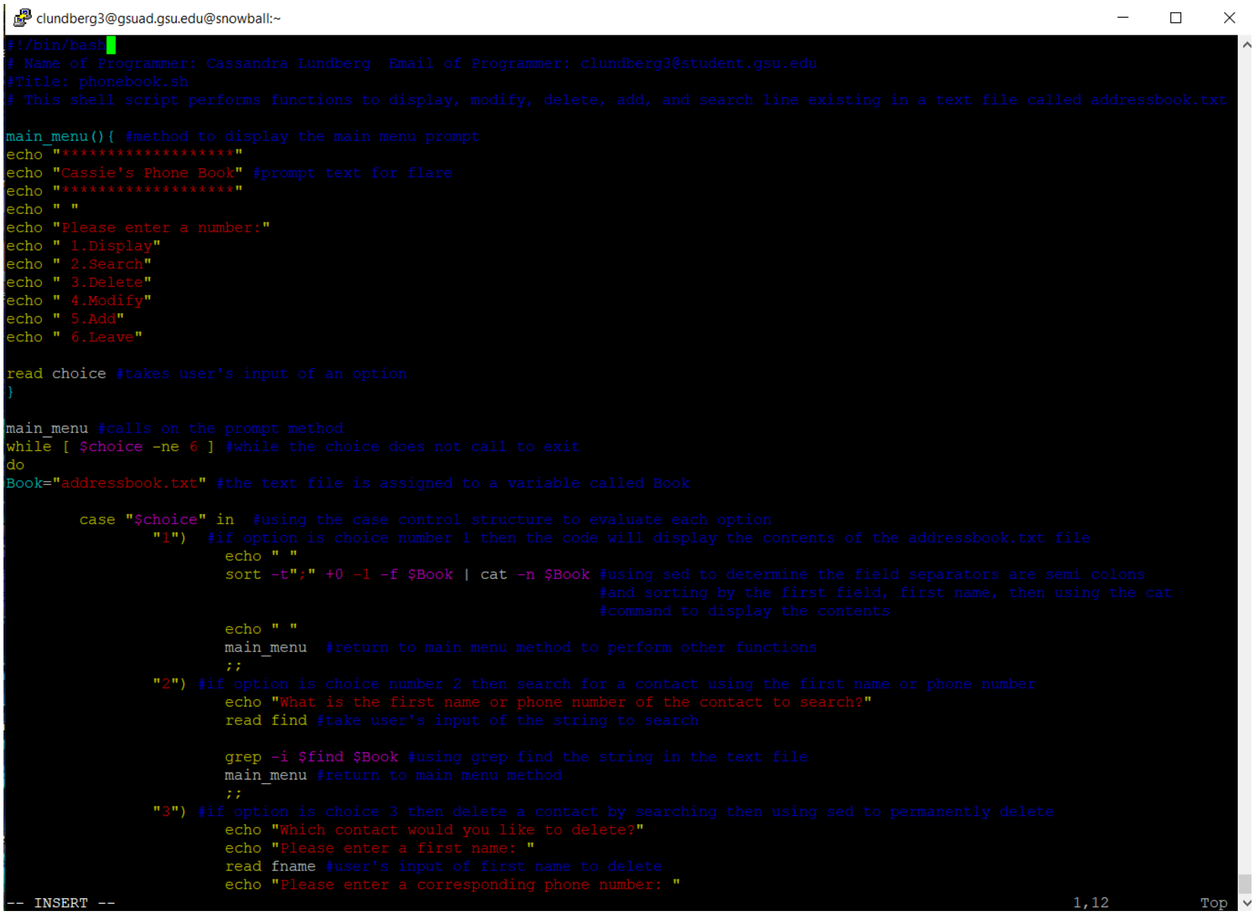
Clear, Cancel, and Exit Functions:



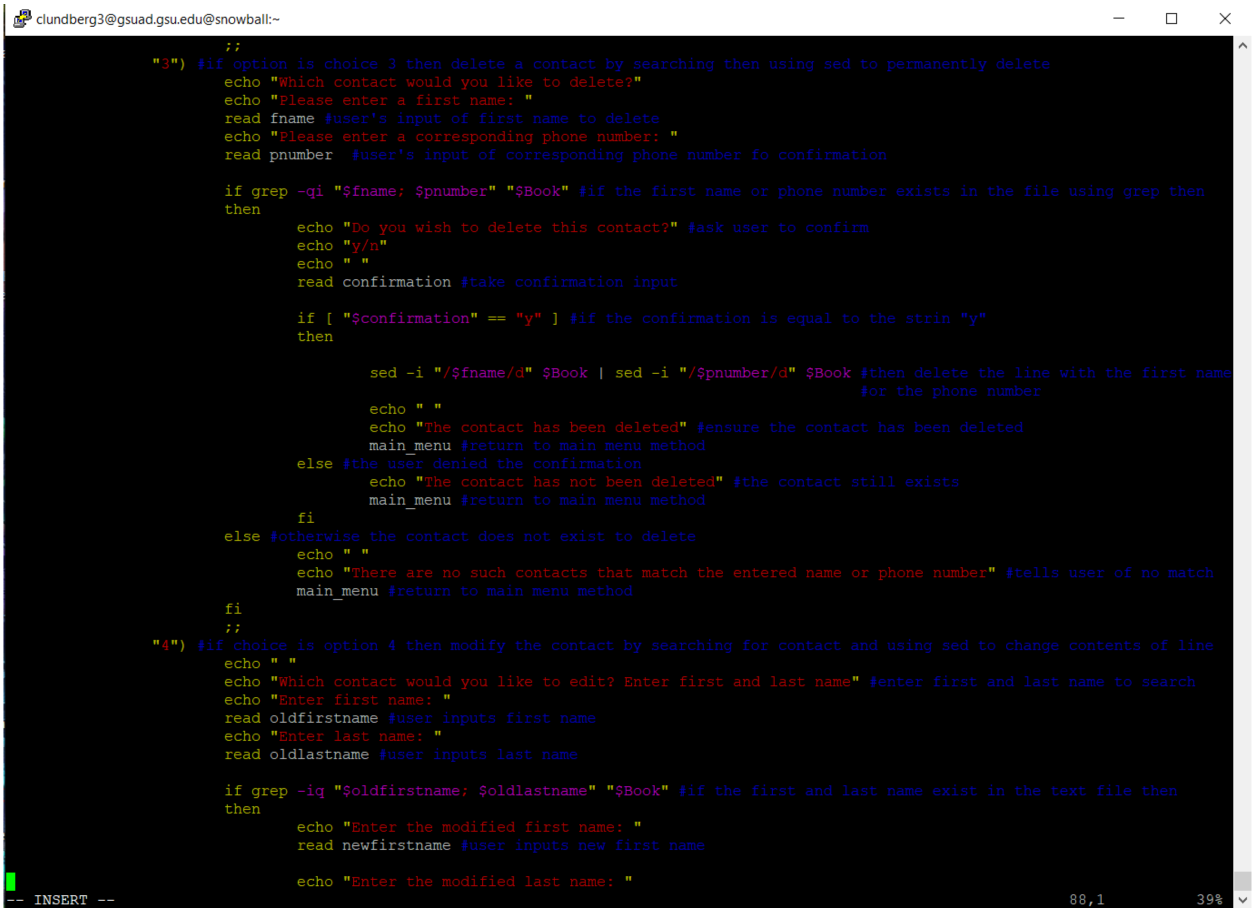
```
Addition (+)
Subtraction (-)
Multiplication (*)
Division (/)
Modulo (%)
Enter exit anytime to exit!
Enter clear to clear entrys | Enter cancel to reenter entry

+
Please enter the first number:

clear
You entered:  clear

Which operation would you like to perform?
Please enter an operation:
Addition (+)
Subtraction (-)
Multiplication (*)
Division (/)
Modulo (%)
Enter exit anytime to exit!
Enter clear to clear entrys | Enter cancel to reenter entry

+
Please enter the first number:

cancel
You entered:  cancel
Please enter the first number:

exit
You entered:  exit
[clundberg3@gsuad.gsu.edu@snowball ~]$
```

## Question: 4

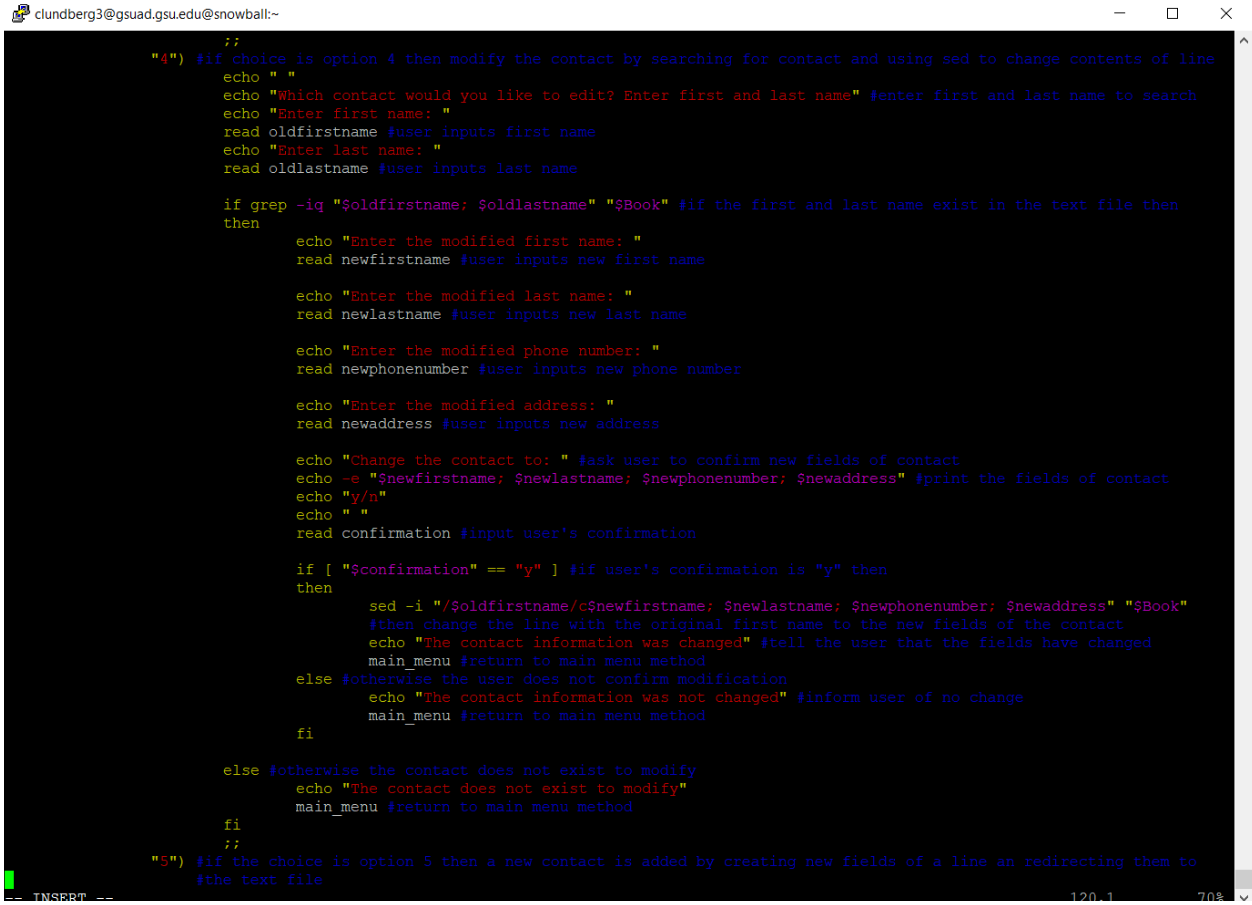
Screenshots of phonebook.sh


clundberg3@gsuad.gsu.edu@snowball:~

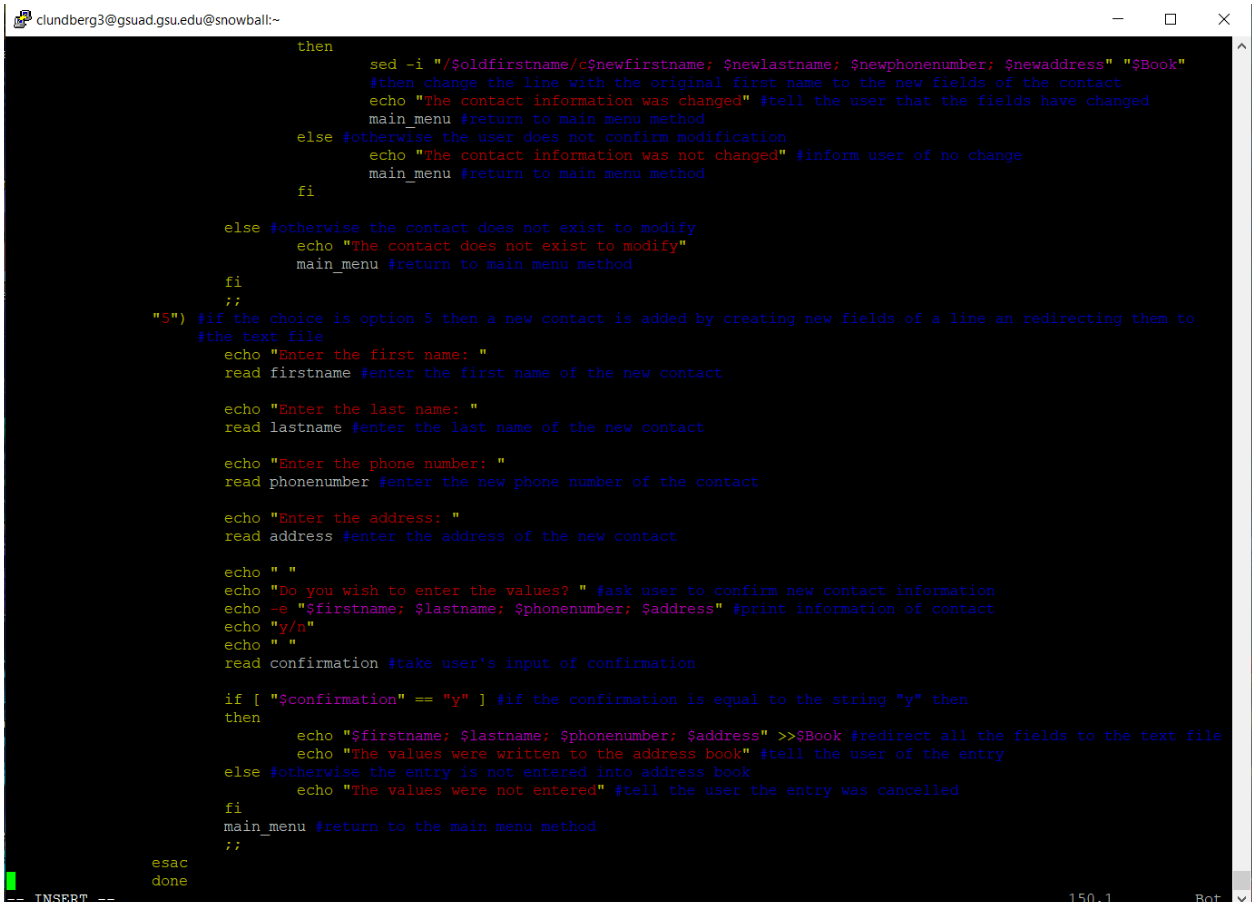
```
#!/bin/bash
# Name of Programmer: Cassandra Lundberg  Email of Programmer: clundberg3@student.gsu.edu
#Title: phonebook.sh
# This shell script performs functions to display, modify, delete, add, and search line existing in a text file called addressbook.txt

main_menu(){ #method to display the main menu prompt
echo "*******************"
echo "Cassie's Phone Book" #prompt text for flare
echo "*******************"
echo " "
echo "Please enter a number:"
echo " 1.Display"
echo " 2.Search"
echo " 3.Delete"
echo " 4.Modify"
echo " 5.Add"
echo " 6.Leave"

read choice #takes user's input of an option
}

main_menu #calls on the prompt method
while [ $choice -ne 6 ] #while the choice does not call to exit
do
Book="addressbook.txt" #the text file is assigned to a variable called Book

        case "$choice" in  #using the case control structure to evaluate each option
                "1")  #if option is choice number 1 then the code will display the contents of the addressbook.txt file
                        echo " "
                        sort -t";" +0 -1 -f $Book | cat -n $Book #using sed to determine the field separators are semi colons
                                                                 #and sorting by the first field, first name, then using the cat
                                                                 #command to display the contents

                        echo " "
                        main_menu  #return to main menu method to perform other functions
                        ;;
                "2") #if option is choice number 2 then search for a contact using the first name or phone number
                        echo "What is the first name or phone number of the contact to search?"
                        read find #take user's input of the string to search

                        grep -i $find $Book #using grep find the string in the text file
                        main_menu #return to main menu method
                        ;;
                "3") #if option is choice 3 then delete a contact by searching then using sed to permanently delete
                        echo "Which contact would you like to delete?"
                        echo "Please enter a first name: "
                        read fname #user's input of first name to delete
                        echo "Please enter a corresponding phone number: "
-- INSERT --                                                                                                 1,12          Top
```



```
                        ;;
                "3") #if option is choice 3 then delete a contact by searching then using sed to permanently delete
                        echo "Which contact would you like to delete?"
                        echo "Please enter a first name: "
                        read fname #user's input of first name to delete
                        echo "Please enter a corresponding phone number: "
                        read pnumber  #user's input of corresponding phone number fo confirmation

                        if grep -qi "$fname; $pnumber" "$Book" #if the first name or phone number exists in the file using grep then
                        then
                                echo "Do you wish to delete this contact?" #ask user to confirm
                                echo "y/n"
                                echo " "
                                read confirmation #take confirmation input

                                if [ "$confirmation" == "y" ] #if the confirmation is equal to the strin "y"
                                then

                                        sed -i "/$fname/d" $Book | sed -i "/$pnumber/d" $Book #then delete the line with the first name
                                                                                              #or the phone number
                                        echo " "
                                        echo "The contact has been deleted" #ensure the contact has been deleted
                                        main_menu #return to main menu method
                                else #the user denied the confirmation
                                        echo "The contact has not been deleted" #the contact still exists
                                        main_menu #return to main menu method
                                fi
                        else #otherwise the contact does not exist to delete
                                echo " "
                                echo "There are no such contacts that match the entered name or phone number" #tells user of no match
                                main_menu #return to main menu method
                        fi
                        ;;
                "4") #if choice is option 4 then modify the contact by searching for contact and using sed to change contents of line
                        echo " "
                        echo "Which contact would you like to edit? Enter first and last name" #enter first and last name to search
                        echo "Enter first name: "
                        read oldfirstname #user inputs first name
                        echo "Enter last name: "
                        read oldlastname #user inputs last name

                        if grep -iq "$oldfirstname; $oldlastname" "$Book" #if the first and last name exist in the text file then
                        then
                                echo "Enter the modified first name: "
                                read newfirstname #user inputs new first name

                                echo "Enter the modified last name: "
-- INSERT --                                                                                                         88,1          39%
```



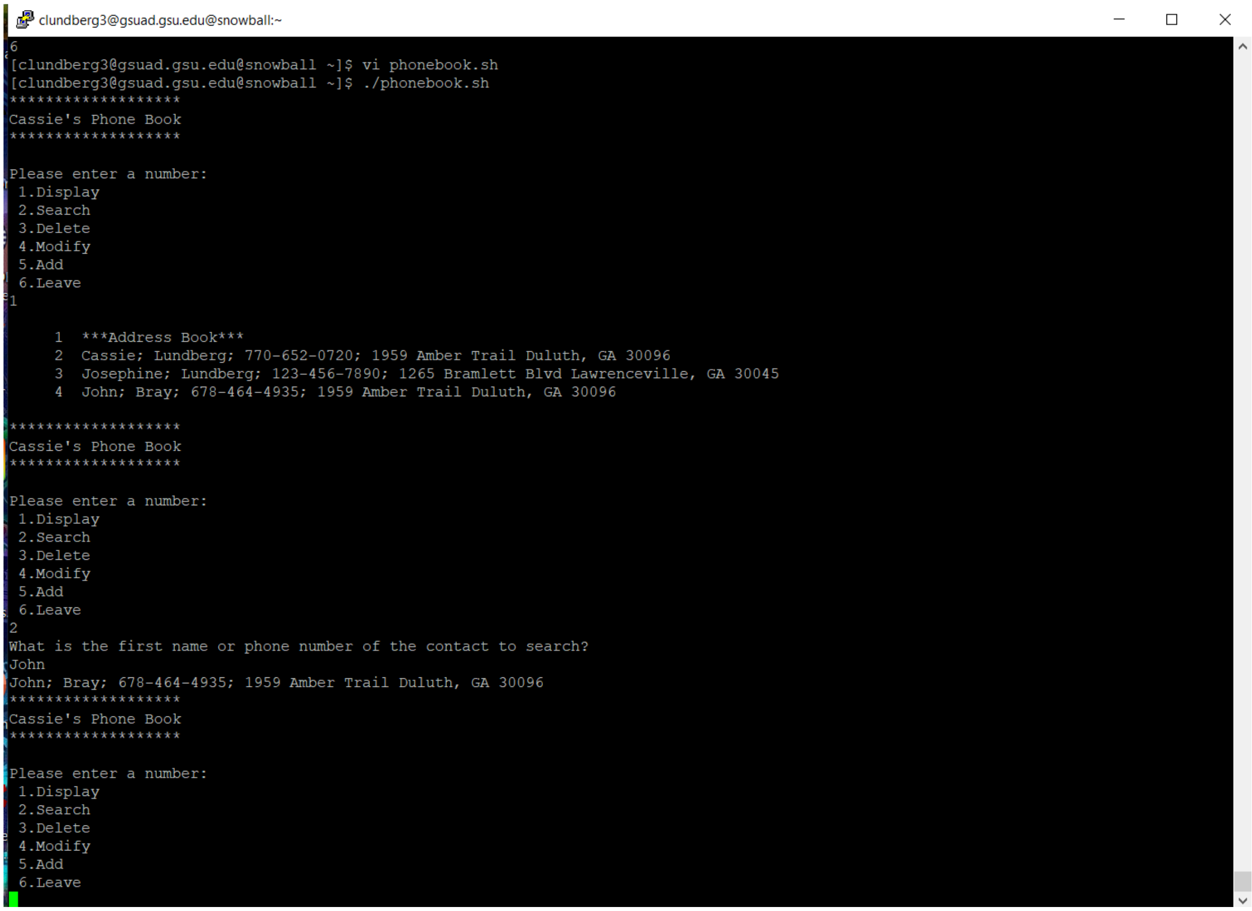
```
                        ;;
                "4") #if choice is option 4 then modify the contact by searching for contact and using sed to change contents of line
                        echo " "
                        echo "Which contact would you like to edit? Enter first and last name" #enter first and last name to search
                        echo "Enter first name: "
                        read oldfirstname #user inputs first name
                        echo "Enter last name: "
                        read oldlastname #user inputs last name

                        if grep -iq "$oldfirstname; $oldlastname" "$Book" #if the first and last name exist in the text file then
                        then
                                echo "Enter the modified first name: "
                                read newfirstname #user inputs new first name

                                echo "Enter the modified last name: "
                                read newlastname #user inputs new last name

                                echo "Enter the modified phone number: "
                                read newphonenumber #user inputs new phone number

                                echo "Enter the modified address: "
                                read newaddress #user inputs new address

                                echo "Change the contact to: " #ask user to confirm new fields of contact
                                echo -e "$newfirstname; $newlastname; $newphonenumber; $newaddress" #print the fields of contact
                                echo "y/n"
                                echo " "
                                read confirmation #input user's confirmation

                                if [ "$confirmation" == "y" ] #if user's confirmation is "y" then
                                then
                                        sed -i "/$oldfirstname/c$newfirstname; $newlastname; $newphonenumber; $newaddress" "$Book"
                                        #then change the line with the original first name to the new fields of the contact
                                        echo "The contact information was changed" #tell the user that the fields have changed
                                        main_menu #return to main menu method
                                else #otherwise the user does not confirm modification
                                        echo "The contact information was not changed" #inform user of no change
                                        main_menu #return to main menu method
                                fi

                        else #otherwise the contact does not exist to modify
                                echo "The contact does not exist to modify"
                                main_menu #return to main menu method
                        fi
                        ;;
                "5") #if the choice is option 5 then a new contact is added by creating new fields of a line an redirecting them to
                     #the text file
-- INSERT --                                                                                                  120,1         70%
```



```
                                then
                                        sed -i "/$oldfirstname/c$newfirstname; $newlastname; $newphonenumber; $newaddress" "$Book"
                                        #then change the line with the original first name to the new fields of the contact
                                        echo "The contact information was changed" #tell the user that the fields have changed
                                        main_menu #return to main menu method
                                else #otherwise the user does not confirm modification
                                        echo "The contact information was not changed" #inform user of no change
                                        main_menu #return to main menu method
                                fi

                        else #otherwise the contact does not exist to modify
                                echo "The contact does not exist to modify"
                                main_menu #return to main menu method
                        fi
                        ;;
                "5") #if the choice is option 5 then a new contact is added by creating new fields of a line an redirecting them to
                     #the text file
                        echo "Enter the first name: "
                        read firstname #enter the first name of the new contact

                        echo "Enter the last name: "
                        read lastname #enter the last name of the new contact

                        echo "Enter the phone number: "
                        read phonenumber #enter the new phone number of the contact

                        echo "Enter the address: "
                        read address #enter the address of the new contact

                        echo " "
                        echo "Do you wish to enter the values? " #ask user to confirm new contact information
                        echo -e "$firstname; $lastname; $phonenumber; $address" #print information of contact
                        echo "y/n"
                        echo " "
                        read confirmation #take user's input of confirmation

                        if [ "$confirmation" == "y" ] #if the confirmation is equal to the string "y" then
                        then
                                echo "$firstname; $lastname; $phonenumber; $address" >>$Book #redirect all the fields to the text file
                                echo "The values were written to the address book" #tell the user of the entry
                        else #otherwise the entry is not entered into address book
                                echo "The values were not entered" #tell the user the entry was cancelled
                        fi
                        main_menu #return to the main menu method
                        ;;
                esac
                done
-- INSERT --                                                                                                   150,1         Bot
```

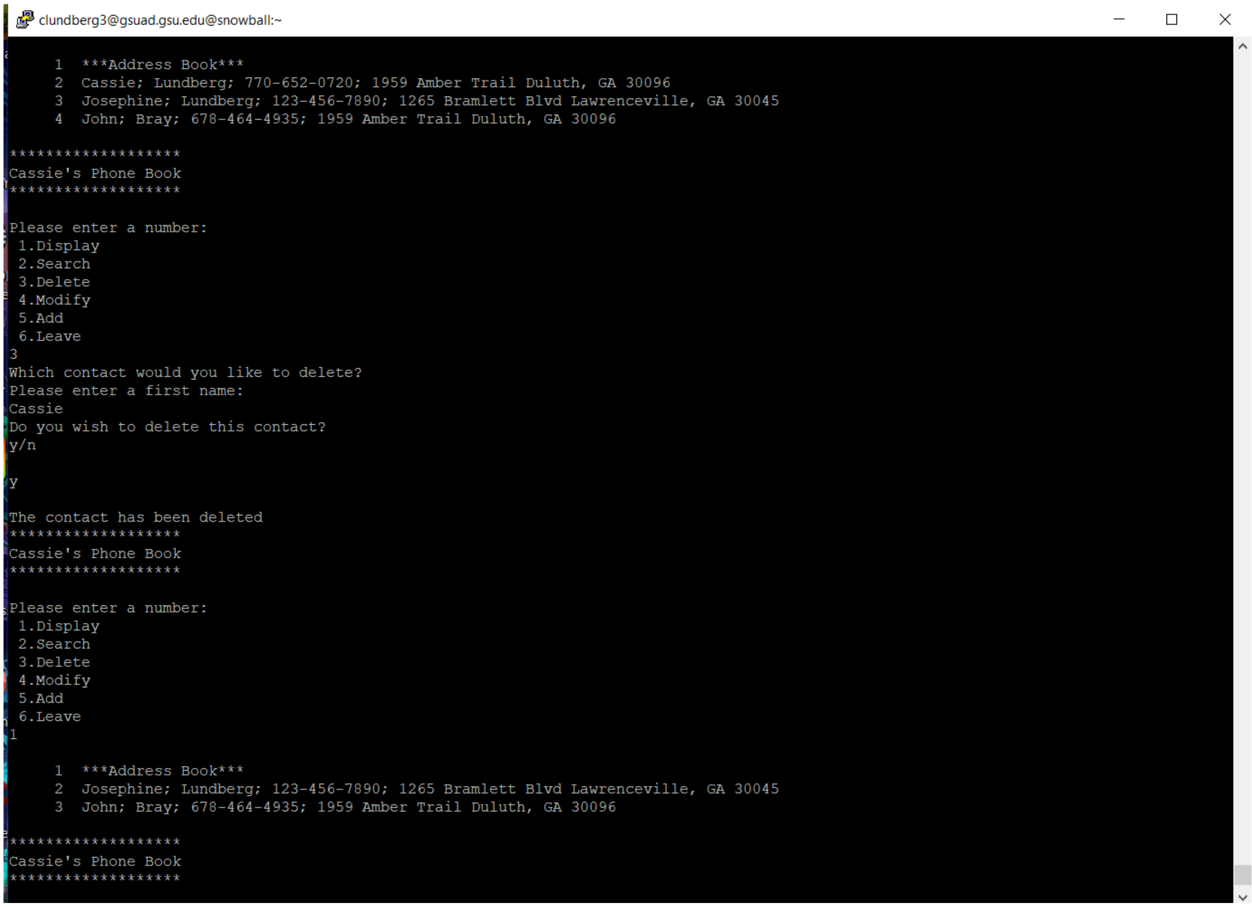
Screenshots of the options:

## Displaying and Searching:



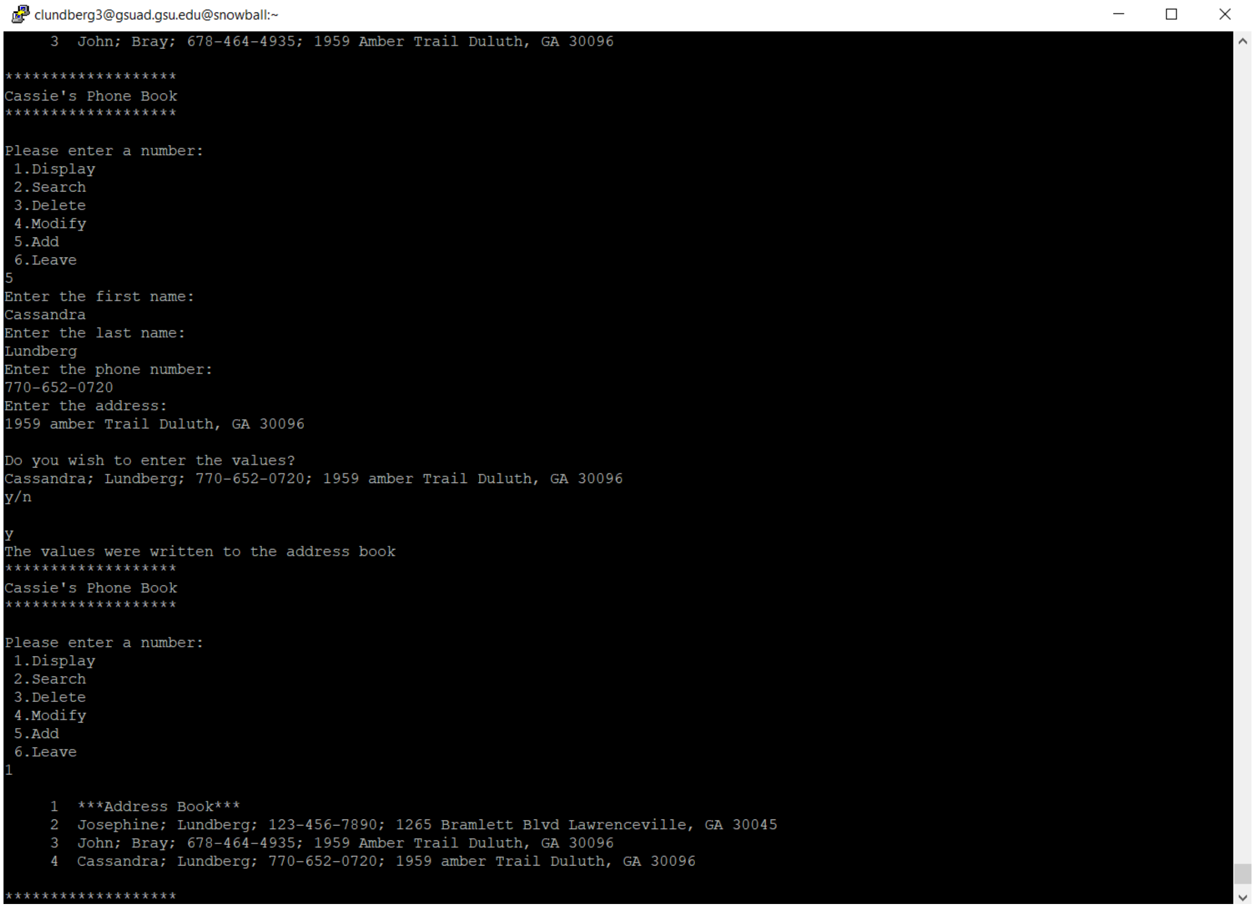
```
6
[clundberg3@gsuad.gsu.edu@snowball ~]$ vi phonebook.sh
[clundberg3@gsuad.gsu.edu@snowball ~]$ ./phonebook.sh
*******************
Cassie's Phone Book
*******************

Please enter a number:
 1.Display
 2.Search
 3.Delete
 4.Modify
 5.Add
 6.Leave
1

     1  ***Address Book***
     2  Cassie; Lundberg; 770-652-0720; 1959 Amber Trail Duluth, GA 30096
     3  Josephine; Lundberg; 123-456-7890; 1265 Bramlett Blvd Lawrenceville, GA 30045
     4  John; Bray; 678-464-4935; 1959 Amber Trail Duluth, GA 30096

*******************
Cassie's Phone Book
*******************

Please enter a number:
 1.Display
 2.Search
 3.Delete
 4.Modify
 5.Add
 6.Leave
2
What is the first name or phone number of the contact to search?
John
John; Bray; 678-464-4935; 1959 Amber Trail Duluth, GA 30096
*******************
Cassie's Phone Book
*******************

Please enter a number:
 1.Display
 2.Search
 3.Delete
 4.Modify
 5.Add
 6.Leave
```

Deleting: Screenshot shows the display before delete, the confirmation of delete and the after display of the delete option.



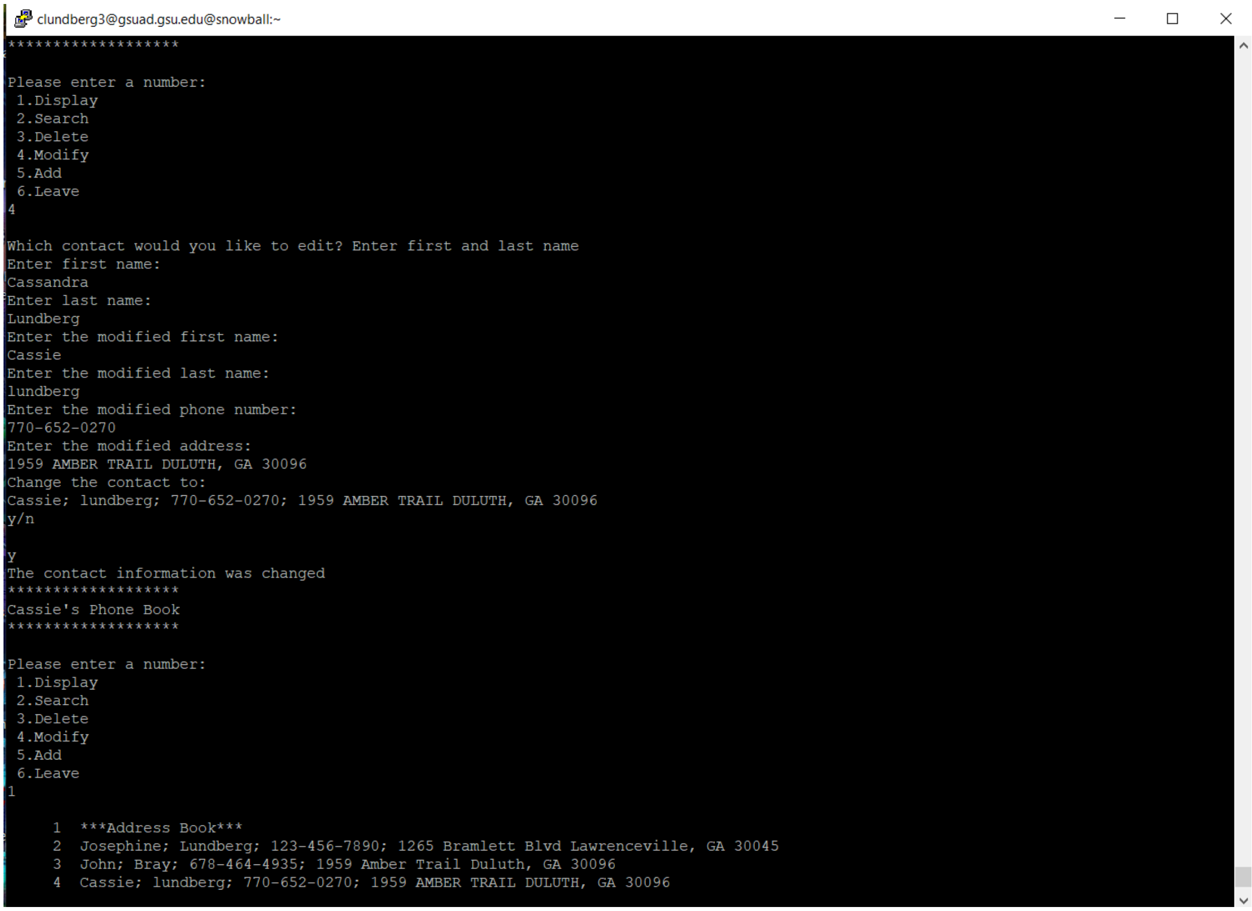
```
     1  ***Address Book***
     2  Cassie; Lundberg; 770-652-0720; 1959 Amber Trail Duluth, GA 30096
     3  Josephine; Lundberg; 123-456-7890; 1265 Bramlett Blvd Lawrenceville, GA 30045
     4  John; Bray; 678-464-4935; 1959 Amber Trail Duluth, GA 30096

*******************
Cassie's Phone Book
*******************

Please enter a number:
 1.Display
 2.Search
 3.Delete
 4.Modify
 5.Add
 6.Leave
3
Which contact would you like to delete?
Please enter a first name:
Cassie
Do you wish to delete this contact?
y/n

y

The contact has been deleted
*******************
Cassie's Phone Book
*******************

Please enter a number:
 1.Display
 2.Search
 3.Delete
 4.Modify
 5.Add
 6.Leave
1

     1  ***Address Book***
     2  Josephine; Lundberg; 123-456-7890; 1265 Bramlett Blvd Lawrenceville, GA 30045
     3  John; Bray; 678-464-4935; 1959 Amber Trail Duluth, GA 30096

*******************
Cassie's Phone Book
*******************
```

Adding: Screenshot shows adding the contact back into the addressbook.txt after deleting it and the display of the contact back in the text file.



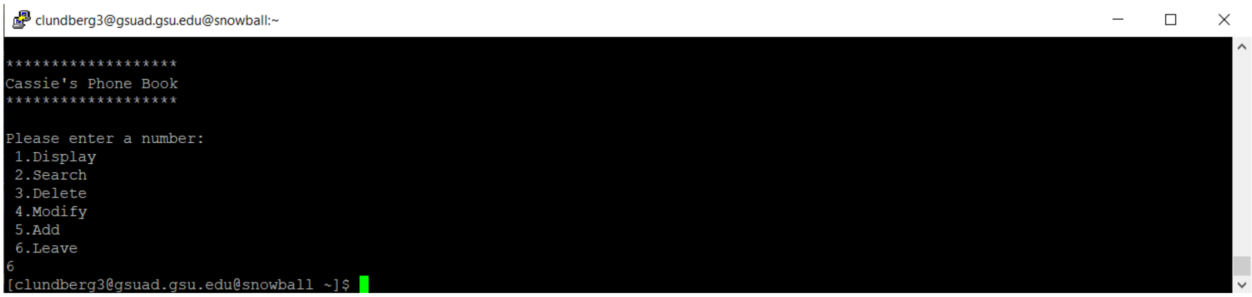
```
     3  John; Bray; 678-464-4935; 1959 Amber Trail Duluth, GA 30096

*******************
Cassie's Phone Book
*******************

Please enter a number:
 1.Display
 2.Search
 3.Delete
 4.Modify
 5.Add
 6.Leave
5
Enter the first name:
Cassandra
Enter the last name:
Lundberg
Enter the phone number:
770-652-0720
Enter the address:
1959 amber Trail Duluth, GA 30096

Do you wish to enter the values?
Cassandra; Lundberg; 770-652-0720; 1959 amber Trail Duluth, GA 30096
y/n

y
The values were written to the address book
*******************
Cassie's Phone Book
*******************

Please enter a number:
 1.Display
 2.Search
 3.Delete
 4.Modify
 5.Add
 6.Leave
1

     1  ***Address Book***
     2  Josephine; Lundberg; 123-456-7890; 1265 Bramlett Blvd Lawrenceville, GA 30045
     3  John; Bray; 678-464-4935; 1959 Amber Trail Duluth, GA 30096
     4  Cassandra; Lundberg; 770-652-0720; 1959 amber Trail Duluth, GA 30096

*******************
```

Modifying: Screenshot shows modifications to the Cassandra contact in changing the first name to Cassie, changing the last name to lundberg, and the phone number to 770-652-0270 and the address to all capital letters.



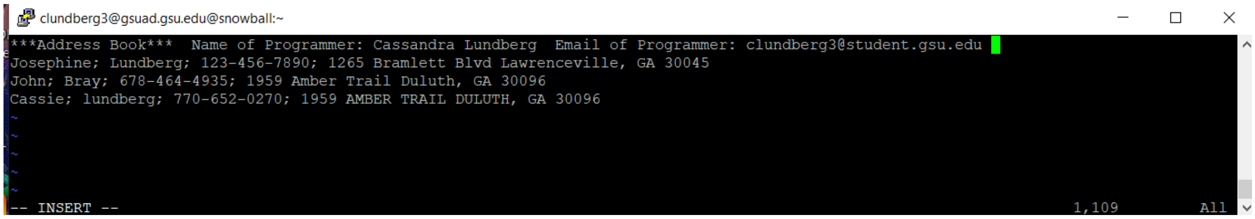
```
******************

Please enter a number:
 1.Display
 2.Search
 3.Delete
 4.Modify
 5.Add
 6.Leave
4

Which contact would you like to edit? Enter first and last name
Enter first name:
Cassandra
Enter last name:
Lundberg
Enter the modified first name:
Cassie
Enter the modified last name:
lundberg
Enter the modified phone number:
770-652-0270
Enter the modified address:
1959 AMBER TRAIL DULUTH, GA 30096
Change the contact to:
Cassie; lundberg; 770-652-0270; 1959 AMBER TRAIL DULUTH, GA 30096
y/n

y
The contact information was changed
******************
Cassie's Phone Book
******************

Please enter a number:
 1.Display
 2.Search
 3.Delete
 4.Modify
 5.Add
 6.Leave
1

     1  ***Address Book***
     2  Josephine; Lundberg; 123-456-7890; 1265 Bramlett Blvd Lawrenceville, GA 30045
     3  John; Bray; 678-464-4935; 1959 Amber Trail Duluth, GA 30096
     4  Cassie; lundberg; 770-652-0270; 1959 AMBER TRAIL DULUTH, GA 30096
```

Leave: Screenshot shows what happens when the leave option (6) is entered:



```

******************
Cassie's Phone Book
******************

Please enter a number:
 1.Display
 2.Search
 3.Delete
 4.Modify
 5.Add
 6.Leave
6
[clundberg3@gsuad.gsu.edu@snowball ~]$ 
```

Screenshot shows addressbook.txt:

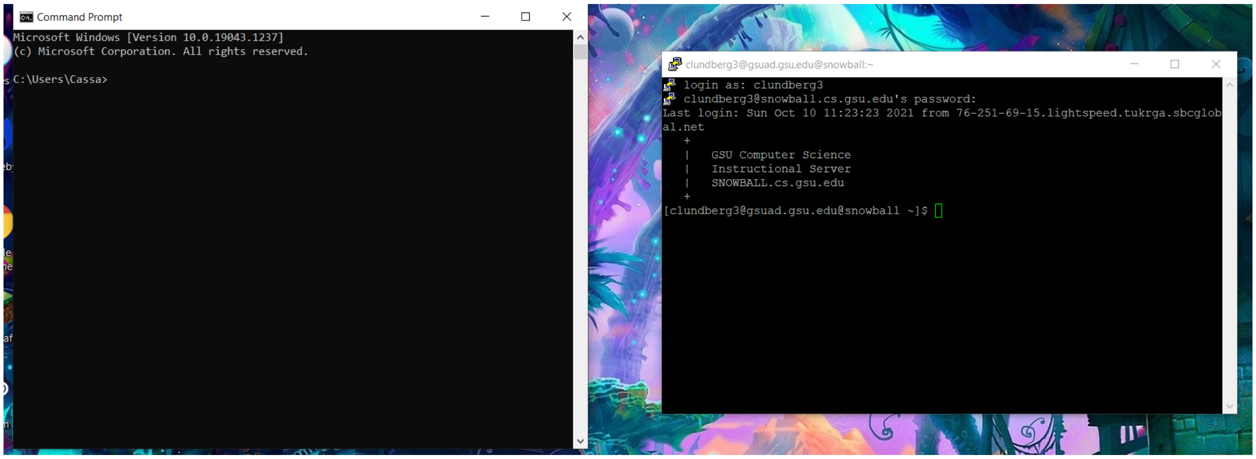


```
***Address Book***  Name of Programmer: Cassandra Lundberg  Email of Programmer: clundberg3@student.gsu.edu
Josephine; Lundberg; 123-456-7890; 1265 Bramlett Blvd Lawrenceville, GA 30045
John; Bray; 678-464-4935; 1959 Amber Trail Duluth, GA 30096
Cassie; lundberg; 770-652-0270; 1959 AMBER TRAIL DULUTH, GA 30096
~
~
~
~
~
-- INSERT --                                                                                 1,109         All
```

## Question: 5

A. Shell is an interface which takes commands input from a keyboard and hands them to the operating system. Since a shell acts as an interpreter between commands from the user and the operating system, it is extremely important just to simply use a computer. Some examples of utilities a shell offers are; file management, programming, writing, editing, and saving text files, and assigning permissions to files.

B.



```
Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Cassa>
```

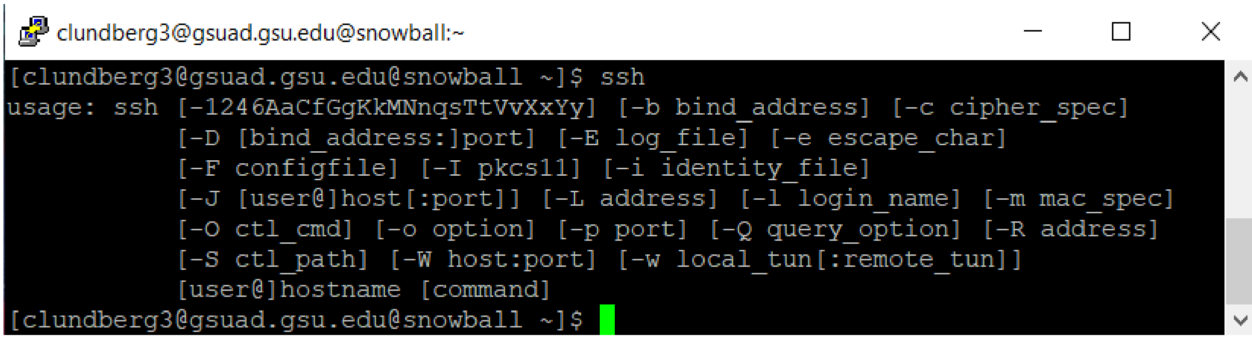
```
login as: clundberg3
clundberg3@snowball.cs.gsu.edu's password:
Last login: Sun Oct 10 11:23:23 2021 from 76-251-69-15.lightspeed.tukrga.sbcglob
al.net
   +
   |   GSU Computer Science
   |   Instructional Server
   |   SNOWBALL.cs.gsu.edu
   +
[clundberg3@gsuad.gsu.edu@snowball ~]$
```

One of the biggest differences between the PC terminal and the snowball server terminal is the starter icon. The PC terminal shows C:\Users\myuser name> then the cursor is blinking. However on the snowball server, one I sign in, the cursor is a “$” to represent that the shell is ready to read a command. This is because Windows uses a PowerShell and the snowball server uses a Bash shell.

C. Since C is a compiled language it requires a compiler to translate the source code from C programming language to a machine language code. C also needs a text editor or code editor for writing and saving C code. For hardware it requires at least CPU and RAM memory as well as input/ output devices such as a keyboard, monitor, and mouse for entering commands.

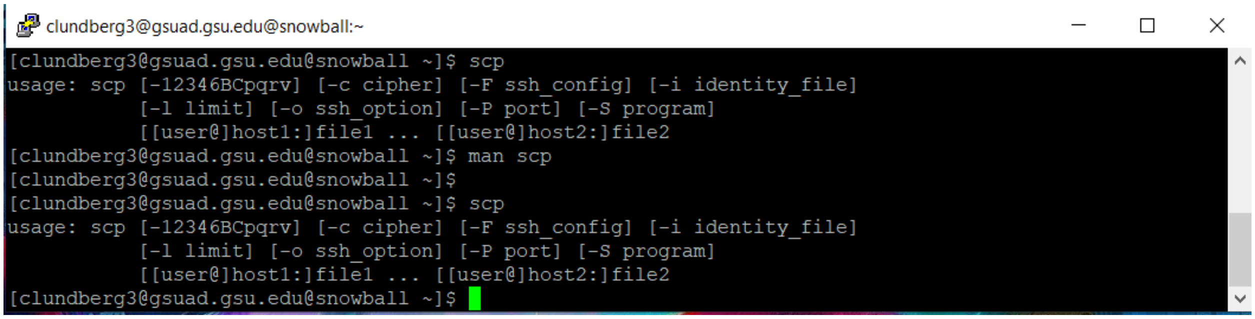
D. As “echo” and “printf()” are both built in commands, they are different in the way where echo always exits with a zero status while printf() can give a non zero exit code status. Printf() also gives more control over the output format and allows for definition of a formatting string. Visually “echo” has a default new line character but “printf()” requires it to manually be added.

E. The “ssh” command is a program for logging into a remote machine and is intended to provide secure encrypted communications between two untrusted hosts over an insecure network.



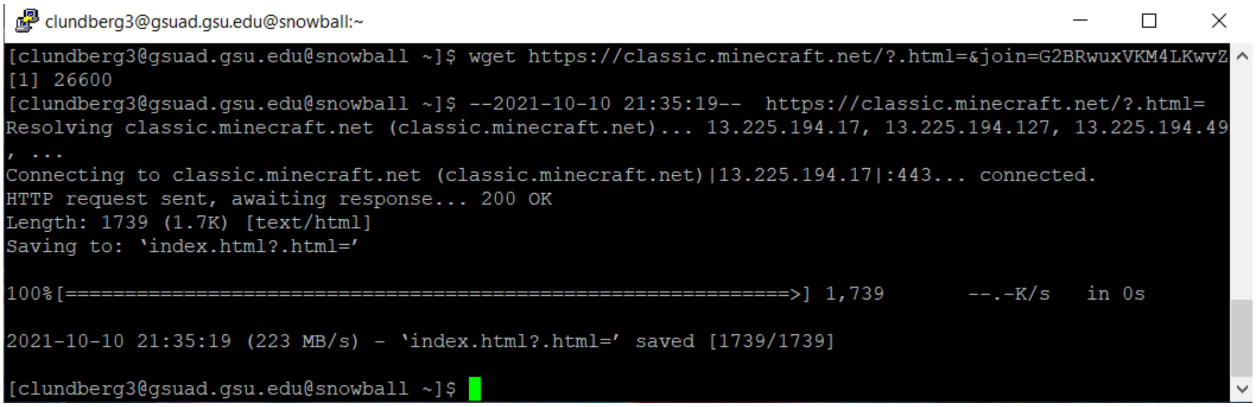
```
[clundberg3@gsuad.gsu.edu@snowball ~]$ ssh
usage: ssh [-1246AaCfGgKkMNnqsTtVvXxYy] [-b bind_address] [-c cipher_spec]
           [-D [bind_address:]port] [-E log_file] [-e escape_char]
           [-F configfile] [-I pkcs11] [-i identity_file]
           [-J [user@]host[:port]] [-L address] [-l login_name] [-m mac_spec]
           [-O ctl_cmd] [-o option] [-p port] [-Q query_option] [-R address]
           [-S ctl_path] [-W host:port] [-w local_tun[:remote_tun]]
           [user@]hostname [command]
[clundberg3@gsuad.gsu.edu@snowball ~]$
```

The “scp” command copies files between hosts on a network and is a tool used by “ssh” network protocol.



```
[clundberg3@gsuad.gsu.edu@snowball ~]$ scp
usage: scp [-12346BCpqrv] [-c cipher] [-F ssh_config] [-i identity_file]
           [-l limit] [-o ssh_option] [-P port] [-S program]
           [[user@]host1:]file1 ... [[user@]host2:]file2
[clundberg3@gsuad.gsu.edu@snowball ~]$ man scp
[clundberg3@gsuad.gsu.edu@snowball ~]$
[clundberg3@gsuad.gsu.edu@snowball ~]$ scp
usage: scp [-12346BCpqrv] [-c cipher] [-F ssh_config] [-i identity_file]
           [-l limit] [-o ssh_option] [-P port] [-S program]
           [[user@]host1:]file1 ... [[user@]host2:]file2
[clundberg3@gsuad.gsu.edu@snowball ~]$
```

The “wget” command is a utility for non-interactive download of files from the web by using the pasted URL from the server.



```
[clundberg3@gsuad.gsu.edu@snowball ~]$ wget https://classic.minecraft.net/?.html=&join=G2BRwuxVKM4LKwvZ
[1] 26600
[clundberg3@gsuad.gsu.edu@snowball ~]$ --2021-10-10 21:35:19--  https://classic.minecraft.net/?.html=
Resolving classic.minecraft.net (classic.minecraft.net)... 13.225.194.17, 13.225.194.127, 13.225.194.49
, ...
Connecting to classic.minecraft.net (classic.minecraft.net)|13.225.194.17|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1739 (1.7K) [text/html]
Saving to: ‘index.html?.html=’

100%[=============================================================>] 1,739       --.-K/s   in 0s

2021-10-10 21:35:19 (223 MB/s) - ‘index.html?.html=’ saved [1739/1739]

[clundberg3@gsuad.gsu.edu@snowball ~]$
```

P. S. This is super cool! I just downloaded a HTML version of the game Minecraft in one of its oldest releases onto the snowball server!!