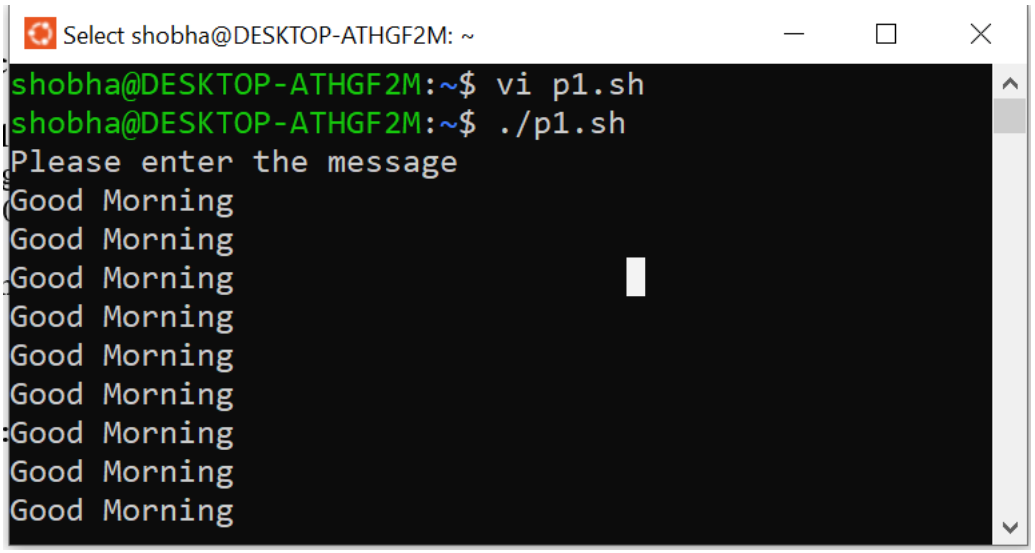
1. a) Write a shell script to read a message "Good Morning" and display it 10 times at regular intervals of 60 seconds.

Shell Script:

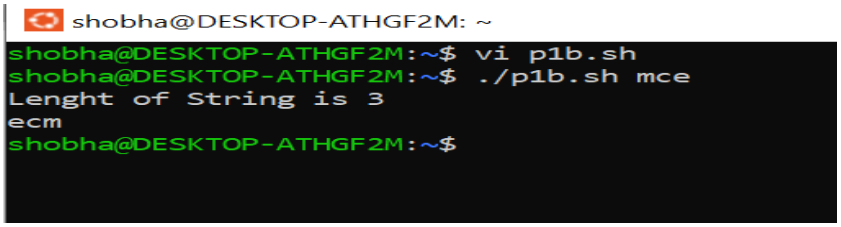
```
echo "Please enter the message"
read msg
for((i=10; i>=1; i--))
do
echo "$msg"
sleep 2
done
```

Output:



1 b) Write a shell script that accepts a string as a command line argument and reverse it.

```
str=$1
b=`echo -n $str|wc -m`
#l=`expr $b - 1`
echo Lenght of String is $b
a=" "
for (( i=b ; i>0 ; i-- ))
do
        c=`echo $str|cut -c $i`
        a=$a$c
done
echo $a
```

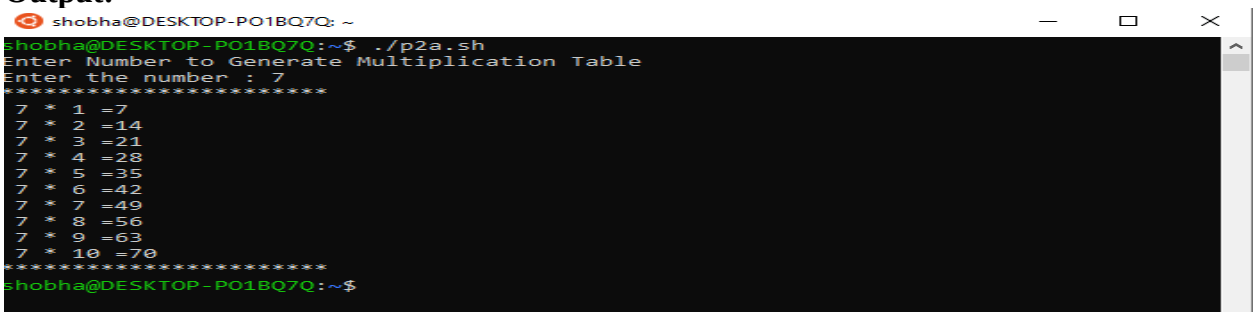


2a. Write a shell script to generate multiplication table.

Shell Script:

```
echo "Enter Number to Generate Multiplication Table"
read -p "Enter the number : " number
echo "************************"
i=1
while [ $i -le 10 ]
do
        echo " $number * $i =`expr $number \* $i ` "
        i=`expr $i + 1`
done
echo "************************"
```

Output:

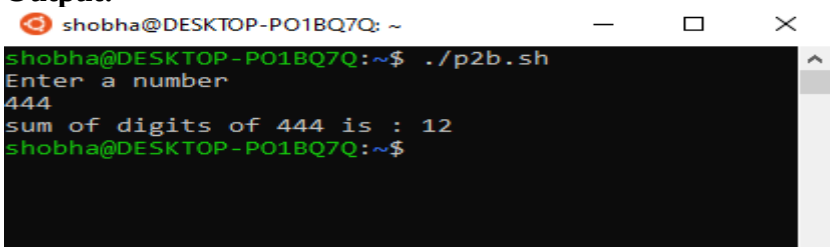


2b. Write a shell script to print sum of individual digits of a number.

Shell Script:

```
echo "Enter a number"
read num
temp=$num
sum=0

while [ $num -gt 0 ]
do
        mod=$((num % 10))    #It will split each digits
           sum=$((sum + mod))   #Add each digit to sum
              num=$((num / 10))    #divide num by 10.
        done

        echo sum of digits of $temp is : $sum
```

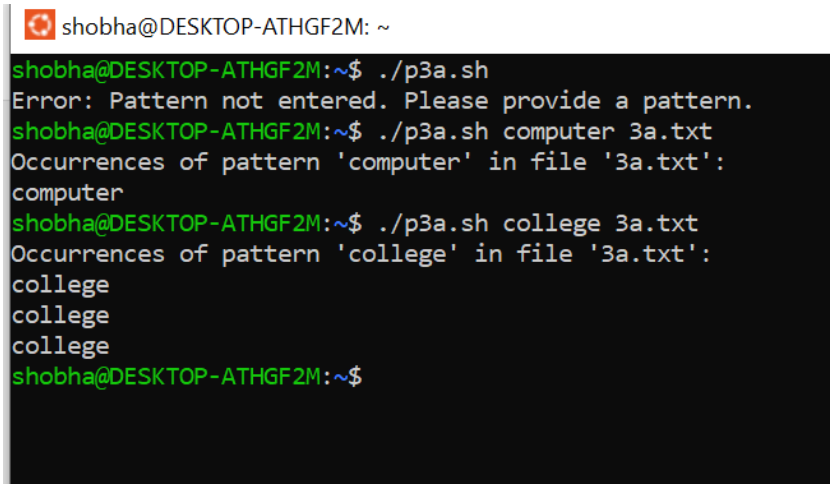
Output:



3a. Write a shell script to search a given pattern in file, if found display the message "Found" or else display "Not found". Accept the pattern and input file as command line arguments

Shell Script:

```
# Check if pattern is provided
if [ -z "$1" ]; then
     echo "Error: Pattern not entered. Please provide a pattern."
          exit 1
fi

# Check if file is provided
if [ -z "$2" ]; then
     echo "Error: File not entered. Please provide a file."
          exit 1
fi

# Assign pattern and file to variables
pattern="$1"
file="$2"

# Perform the grep operation
result=$(grep -o -i "$pattern" "$file")

# Display the result
if [ -z "$result" ]; then
     echo "Pattern '$pattern' not found in file '$file'."
else
     echo "Occurrences of pattern '$pattern' in file '$file':"

   echo "$result"
fi
```



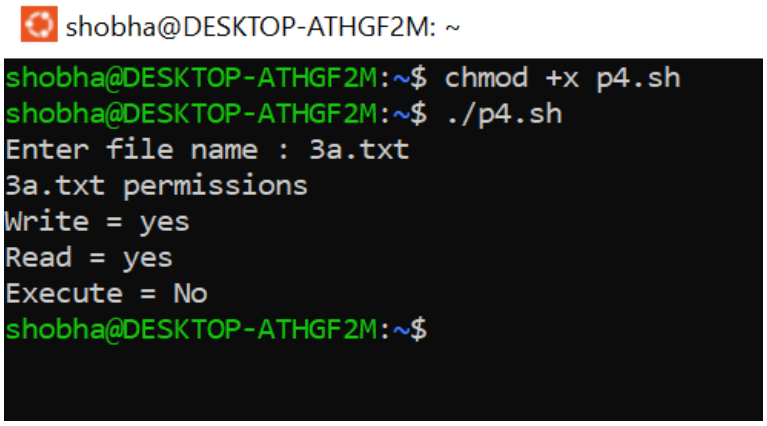
b) Write a shell script to accept the pattern and file to be used. If the pattern is not entered display a message ”String not entered”. If file name is not mentioned display appropriate message.

```
# Check if pattern is provided
if [ -z "$1" ]; then
        echo "Error: Pattern not entered. Please provide a pattern."
           exit 1
fi

# Check if file is provided
if [ -z "$2" ]; then
        echo "Error: File not entered. Please provide a file."
           exit 1
fi

# Assign pattern and file to variables
pattern="$1"
file="$2"

# Perform the grep operation
result=$(grep -o -i "$pattern" "$file")

# Display the result
if [ -z "$result" ]; then
        echo "Pattern '$pattern' not found in file '$file'."
   else
           echo "Occurrences of pattern '$pattern' in file '$file':"
              echo "$result"
fi
```

```
shobha@DESKTOP-ATHGF2M: ~
shobha@DESKTOP-ATHGF2M:~$ ./p3b.sh
Error: Pattern not entered. Please provide a pattern.
shobha@DESKTOP-ATHGF2M:~$ ./p3b.sh college 3a.txt
Occurrences of pattern 'college' in file '3a.txt':
college
college
college
shobha@DESKTOP-ATHGF2M:~$ ./p3b.sh computer 3a.txt
Occurrences of pattern 'computer' in file '3a.txt':
computer
shobha@DESKTOP-ATHGF2M:~$ vi 3a.txt
shobha@DESKTOP-ATHGF2M:~$ cat 3a.txt
computer science and engineering college
malnad college
engineering college
shobha@DESKTOP-ATHGF2M:~$
```

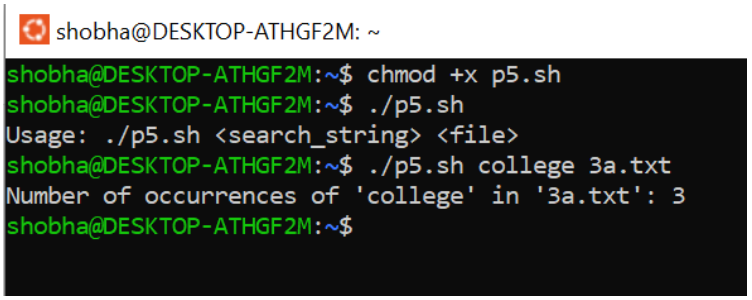
4. Write a shell script to check whether the given file as read and write and execute permission.

```
echo -n "Enter file name : "
read file

# find out if file has write permission or not
[ -w $file ] && W="Write = yes" || W="Write = No"

# find out if file has execute permission or not
[ -x $file ] && X="Execute = yes" || X="Execute = No"

# find out if file has read permission or not
[ -r $file ] && R="Read = yes" || R="Read = No"

echo "$file permissions"
echo "$W"
echo "$R"
echo "$X"
```



5. Write a shell script that searches a given string in a given file and prints the number of times it repeats, else display proper error message. The script should accept the file as command line argument.

```
#!/bin/bash
# Check if the number of command-line arguments is correct
if [ "$#" -ne 2 ]; then
    echo "Usage: $0 <search_string> <file>"
    exit 1
fi

# Assign command-line arguments to variables
search_string="$1"
file="$2"

# Check if the file exists
if [ ! -f "$file" ]; then
    echo "Error: File '$file' not found."
    exit 1
fi


# Use grep to search for the string in the file and count occurrences
count=$(grep -o -i "$search_string" "$file" | wc -l)

# Display the result
echo "Number of occurrences of '$search_string' in '$file': $count"
```



6. Write a shell script to display all the process running in the system every 30 seconds for 5 times using a) while b) for

Using a while loop:

```
#!/bin/bash

count=1

while [ $count -le 5 ]; do
   echo "Displaying all processes (Run #$count):"
   ps aux
   sleep 30
   ((count++))
done
```

Using a for loop:

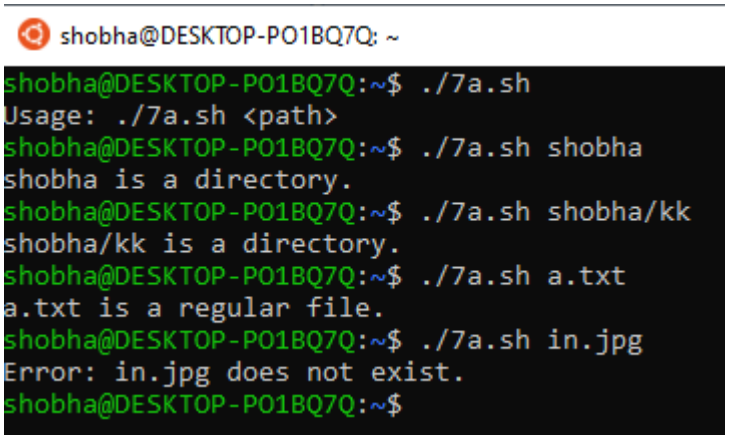
```
#!/bin/bash

for ((count=1; count<=5; count++)); do
   echo "Displaying all processes (Run #$count):"
   ps aux
   sleep 30
done
```

shobha@DESKTOP-ATHGF2M: ~

```
shobha@DESKTOP-ATHGF2M:~$ ./p6b.sh
Displaying all processes (Run #1):
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0   8972   432 ?        Ssl  21:47   0:00 /init
root          12  0.0  0.0   9320   244 tty1     Ss   21:47   0:00 /init
shobha        13  0.0  0.0  14248  3808 tty1     S    21:47   0:00 -bash
shobha       127  0.0  0.0  12820  1780 tty1     S    22:21   0:00 /bin/bash ./p6b.sh
shobha       128  0.0  0.0  15520  1932 tty1     R    22:21   0:00 ps aux
Displaying all processes (Run #2):
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0   8972   432 ?        Ssl  21:47   0:00 /init
root          12  0.0  0.0   9320   244 tty1     Ss   21:47   0:00 /init
shobha        13  0.0  0.0  14248  3808 tty1     S    21:47   0:00 -bash
shobha       127  1.0  0.0  12952  2104 tty1     S    22:21   0:00 /bin/bash ./p6b.sh
shobha       130  0.0  0.0  15520  1932 tty1     R    22:21   0:00 ps aux
Displaying all processes (Run #3):
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0   8972   432 ?        Ssl  21:47   0:00 /init
root          12  0.0  0.0   9320   244 tty1     Ss   21:47   0:00 /init
shobha        13  0.0  0.0  14248  3808 tty1     S    21:47   0:00 -bash
shobha       127  0.6  0.0  12952  2104 tty1     S    22:21   0:00 /bin/bash ./p6b.sh
shobha       132  0.0  0.0  15520  1932 tty1     R    22:21   0:00 ps aux
Displaying all processes (Run #4):
USER         PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root           1  0.0  0.0   8972   432 ?        Ssl  21:47   0:00 /init
root          12  0.0  0.0   9320   244 tty1     Ss   21:47   0:00 /init
shobha        13  0.0  0.0  14248  3808 tty1     S    21:47   0:00 -bash
shobha       127  0.4  0.0  12952  2104 tty1     S    22:21   0:00 /bin/bash ./p6b.sh
shobha       134  0.0  0.0  15520  1928 tty1     R    22:21   0:00 ps aux
Displaying all processes (Run #5):
```

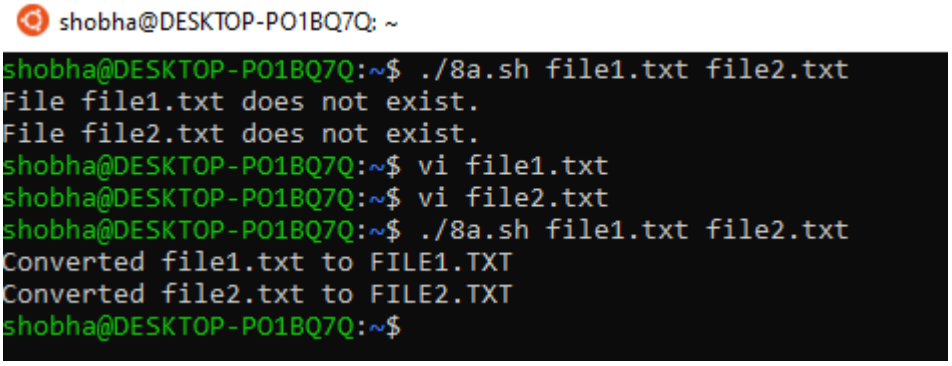
7. Write a shell script that takes a command –line argument and reports on whether it is directory, a file, or something else

```
# Check if the user provided an argument
if [ $# -eq 0 ]; then
  echo "Usage: $0 <path>"
  exit 1
fi

# Get the argument
path=$1

# Check if the path exists
if [ ! -e "$path" ]; then
  echo "Error: $path does not exist."
  exit 1
fi

# Check if it's a directory
if [ -d "$path" ]; then
  echo "$path is a directory."
# Check if it's a regular file
elif [ -f "$path" ]; then
  echo "$path is a regular file."
else
  echo "$path is something else (not a directory or a regular file)."
fi
```



8. Write a shell script that accepts one or more file name as arguments and converts all of them to uppercase, provided they exist in the current directory

```
# Check if at least one argument is provided
if [ $# -eq 0 ]; then
  echo "Usage: $0 <file1> [file2 file3 ...]"
  exit 1
fi

# Loop through each provided file name
for file in "$@"; do
  # Check if the file exists in the current directory
  if [ -e "$file" ]; then
    # Convert the file name to uppercase and rename the file
    new_name=$(echo "$file" | tr '[:lower:]' '[:upper:]')
    mv "$file" "$new_name"
    echo "Converted $file to $new_name"
  else
    echo "File $file does not exist."
  fi
done
```



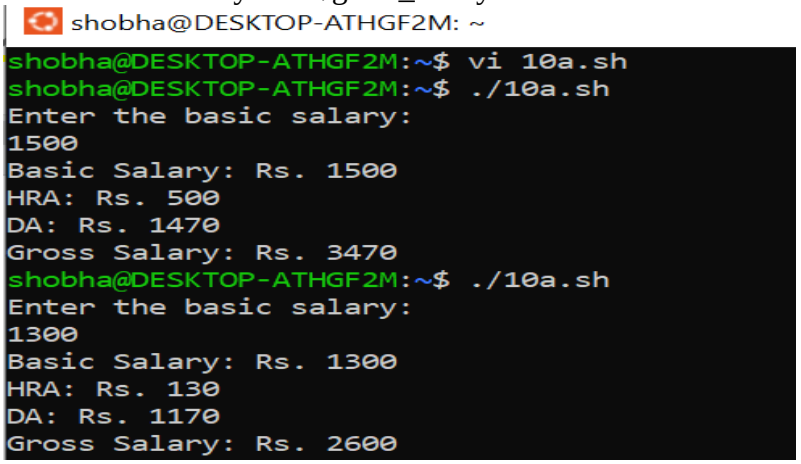
9. Write a shell script that deletes all lines containing a specified word in one or more files supplied as arguments to it

```
# Prompt user for the word to search for
echo "Enter the word to search for in all lines:"
read -r word

# Check if the correct number of arguments is provided
if [ "$#" -eq 0 ]; then
    echo "Usage: $0 <file1> [file2 ...]"
    exit 1
fi

echo "The file names are: $*"

# Loop through all files provided as arguments
for file in "$@"; do
    # Check if the file exists
    if [ ! -e "$file" ]; then
        echo "File '$file' not found. Skipping..."
        continue
    fi

    echo "Lines in '$file' not containing '$word':"
    grep -v "$word" "$file"
done
```

```
shobha@DESKTOP-PO1BQ7Q: ~
shobha@DESKTOP-PO1BQ7Q:~$ cat k1.txt
no one is great
no one is waste
without work nothing is possible
with hard work nothing is impossible
so do your work perfectly without fail

shobha@DESKTOP-PO1BQ7Q:~$ cat k2.txt
 how are you
tell me about your self
how is the day.
shobha@DESKTOP-PO1BQ7Q:~$ ./9b.sh how k1.txt k2.txt
Enter the word to search for in all lines:
how
The file names are: how k1.txt k2.txt
File 'how' not found. Skipping...
Lines in 'k1.txt' not containing 'how':
no one is great
no one is waste
without work nothing is possible
with hard work nothing is impossible
so do your work perfectly without fail

Lines in 'k2.txt' not containing 'how':
tell me about your self
shobha@DESKTOP-PO1BQ7Q:~$
```

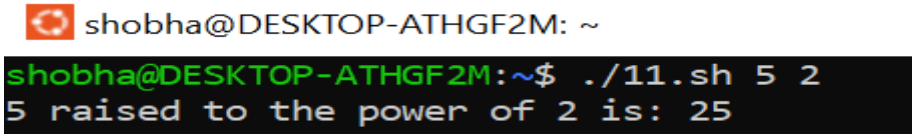
10. Write a shell script that computes the gross salary of an employee according to the following rules:
i) If basic salary is < 1500 then HRA =10% of the basic and DA =90% of the basic.
ii) If basic salary is >=1500 then HRA =Rs500 and DA=98% of the basic the basic salary is entered interactively through the key board.

```
#!/bin/bash
# read basic salary from user
echo "Enter the basic salary: "
read basic_salary

# Check conditions and calculate HRA and DA accordingly
if [ $basic_salary -lt 1500 ]; then
        HRA=`expr $basic_salary \* 10 / 100`
          DA=`expr $basic_salary \* 90 / 100`
    else
            HRA=500
              DA=`expr $basic_salary \* 98 / 100`
fi

# Calculate gross salary
gross_salary=$((basic_salary + HRA + DA))

# Display the results
echo "Basic Salary: Rs. $basic_salary"
echo "HRA: Rs. $HRA"
echo "DA: Rs. $DA"
echo "Gross Salary: Rs. $gross_salary"
```



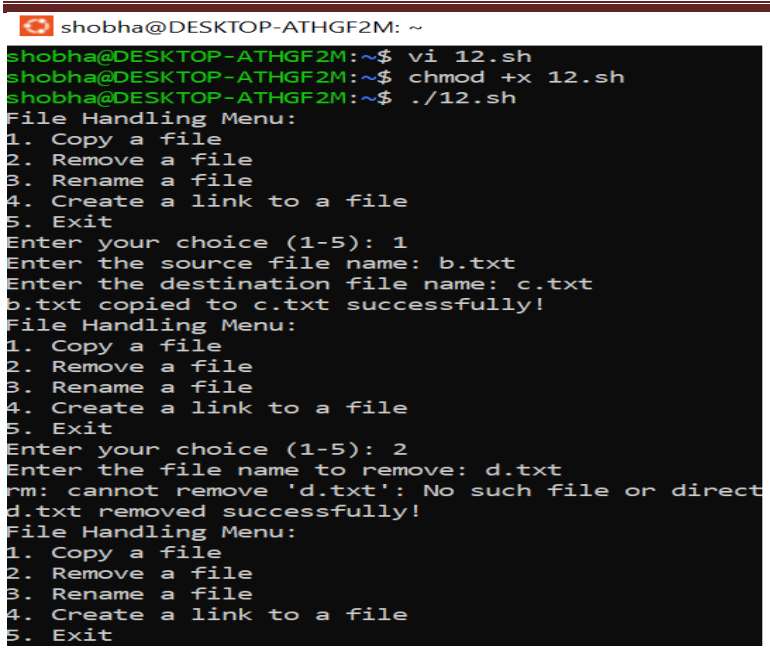
11. Write a shell script that integers as its arguments and computes the value of first number raised to the accepts two power of the second number

```
#!/bin/bash

# Check if two arguments are provided
if [ $# -ne 2 ]; then
    echo "Usage: $0 <base> <exponent>"
    exit 1
fi

# Assign command line arguments to variables
base=$1
exponent=$2

# Use bc to calculate the power
result=$(echo "$base^$exponent" | bc)

# Display the result
echo "$base raised to the power of $exponent is: $result"
```



12. Write an interactive file-handling shell program. Let it offer the user the choice of copying, removing, renaming, or linking files. Once the user has made a choice, have the program ask the User for the necessary information, such as the file name, new name and so on.

```
#!/bin/bash

while true; do
   echo "File Handling Menu:"
   echo "1. Copy a file"
   echo "2. Remove a file"
   echo "3. Rename a file"
   echo "4. Create a link to a file"
   echo "5. Exit"

   read -p "Enter your choice (1-5): " choice

   case $choice in
      1)
         read -p "Enter the source file name: " source_file
         read -p "Enter the destination file name: " destination_file
         cp "$source_file" "$destination_file"
         echo "$source_file copied to $destination_file successfully!"
         ;;
      2)
         read -p "Enter the file name to remove: " file_to_remove
         rm "$file_to_remove"
         echo "$file_to_remove removed successfully!"
         ;;
      3)
         read -p "Enter the current file name: " old_name
         read -p "Enter the new file name: " new_name
         mv "$old_name" "$new_name"
         echo "$old_name renamed to $new_name successfully!"
         ;;
      4)
         read -p "Enter the source file name: " source_file
         read -p "Enter the link name: " link_name
         ln "$source_file" "$link_name"
         echo "Link created successfully! $link_name points to $source_file"
         ;;
      5)
         echo "Exiting program. Goodbye!"
         exit 0
         ;;
      *)
         echo "Invalid choice. Please enter a number between 1 and 5."
         ;;
   esac
done
```


shobha@DESKTOP-ATHGF2M: ~
shobha@DESKTOP-ATHGF2M:~$ vi 12.sh
shobha@DESKTOP-ATHGF2M:~$ chmod +x 12.sh
shobha@DESKTOP-ATHGF2M:~$ ./12.sh
File Handling Menu:
1. Copy a file
2. Remove a file
3. Rename a file
4. Create a link to a file
5. Exit
Enter your choice (1-5): 1
Enter the source file name: b.txt
Enter the destination file name: c.txt
b.txt copied to c.txt successfully!
File Handling Menu:
1. Copy a file
2. Remove a file
3. Rename a file
4. Create a link to a file
5. Exit
Enter your choice (1-5): 2
Enter the file name to remove: d.txt
rm: cannot remove 'd.txt': No such file or direct
d.txt removed successfully!
File Handling Menu:
1. Copy a file
2. Remove a file
3. Rename a file
4. Create a link to a file
5. Exit

13. Write shell script that takes a login name as command – line argument and reports when that Person logs in.

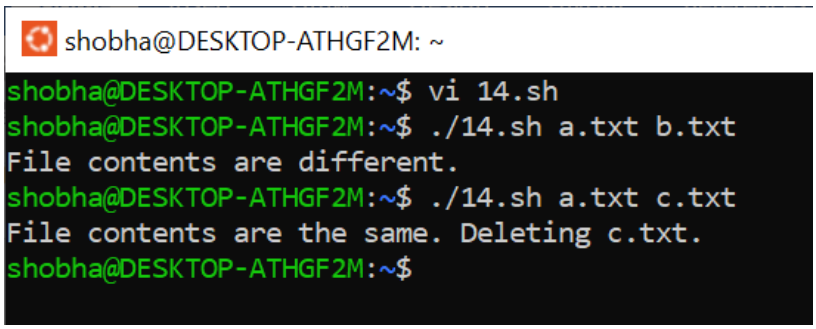
```
#!/bin/bash

# Check if a login name is provided as a command-line argument
if [ "$#" -ne 1 ]; then
        echo "Usage: $0 <login_name>"
           exit 1
fi

login_name=$1

# Check last login information for the specified user
last_login_info=$(last "$login_name")

echo "$last_login_info"
```

Output

```
Enter the user name:
CSESTAFF
CSESTAFF pts/1 192.168.1.245 Sat Jan 31 16:57 still logged in
CSESTAFF pts/0 192.168.1.245 Sat Jan 31 15:41 still logged in
CSESTAFF pts/0 192.168.1.118 Tue Jan 27 15:53 - 16:50 (00:56)
wtmp begins Thu Jan 22 16:39:05 2015
these are the datails of user CSESTAFF
```

14. Write a shell script which receives two file names as arguments. It should check whether the two file contents are same or not. If they are same then second file should be deleted.

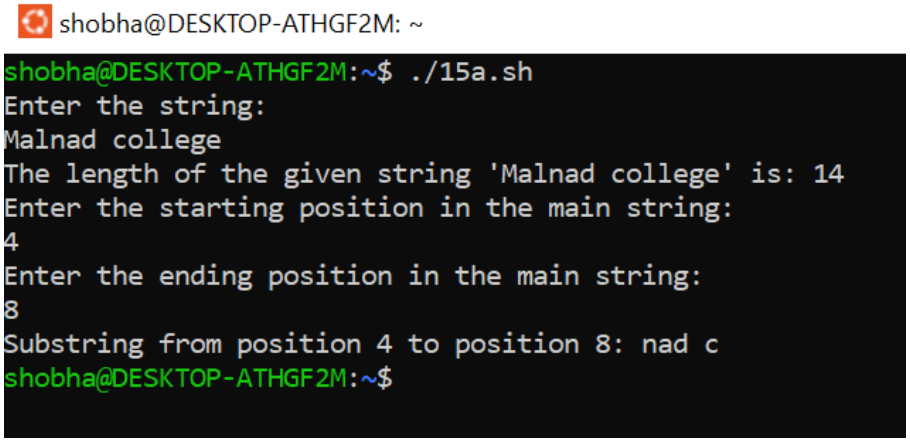
```
#!/bin/bash

# Check if two file names are provided as command-line arguments
if [ "$#" -ne 2 ]; then
    echo "Usage: $0 <file1> <file2>"
    exit 1
fi

file1="$1"
file2="$2"

# Check if the files exist
if [ ! -e "$file1" ] || [ ! -e "$file2" ]; then
    echo "Error: Both files must exist."
    exit 1
fi

# Compare the contents of the two files
if cmp -s "$file1" "$file2"; then
    echo "File contents are the same. Deleting $file2."
    rm "$file2"
else
    echo "File contents are different."
fi
```



15 Write a shell script to perform the following string operations:
i. To extract a sub-string from a given string.
ii. To find the length of a given string.

```
#!/bin/bash

echo "Enter the string:"
read str

# find the length of a given string
strlen=${#str}

echo "The length of the given string '$str' is: $strlen"

echo "Enter the starting position in the main string:"
read s1

echo "Enter the ending position in the main string:"
read f1

# Using cut to extract the substring
substring=$(echo "$str" | cut -c "$s1"-"$f1")
echo "Substring from position $s1 to position $f1: $substring"
```

Output

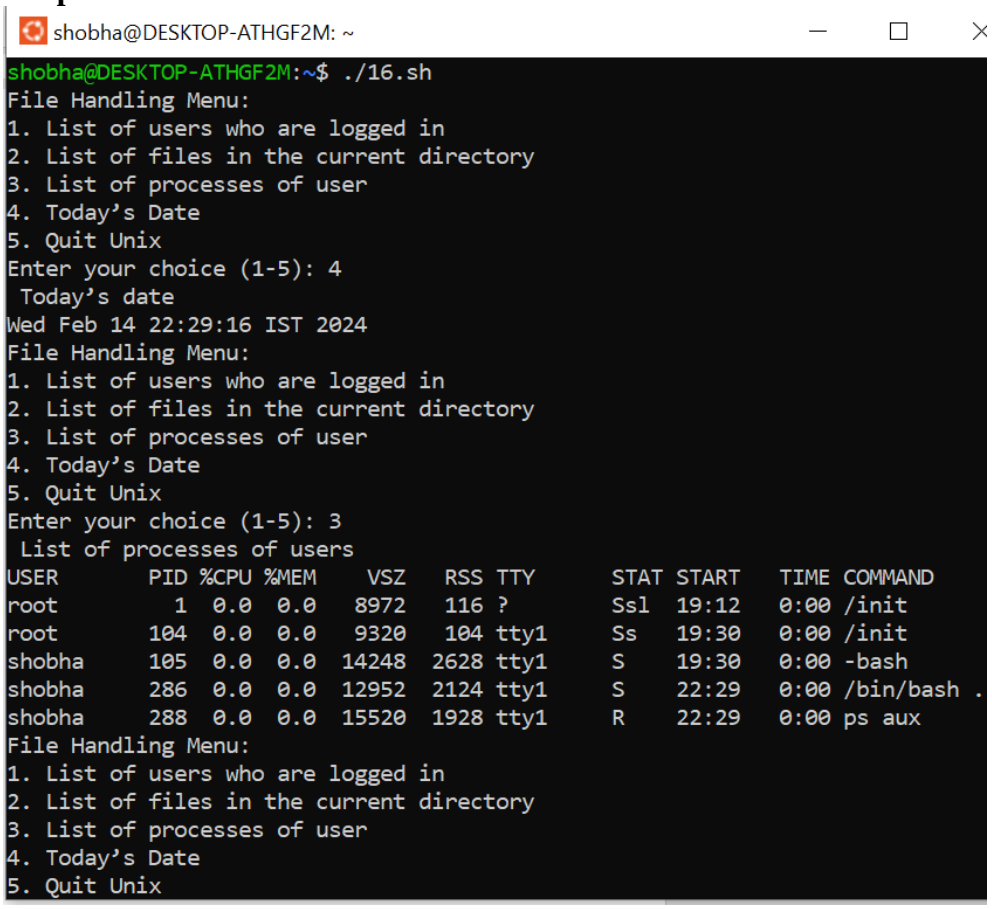


16 Write a menu driven shell script to perform the following:

i) List of users who are logged in
ii) List of files in the current directory
iii) List of processes of users
iv) Today's date
v) Quit to Unix

```
#!/bin/bash
while true; do
   echo "File Handling Menu:"
   echo "1. List of users who are logged in"
   echo "2. List of files in the current directory"
   echo "3. List of processes of user"
   echo "4. Today's Date"
   echo "5. Quit Unix"

   read -p "Enter your choice (1-5): " choice

   case $choice in
      1)
         echo "List of Users who have logged in"
            who
         ;;
      2)
          echo "List of files in current directory"
             ls -l
         ;;
      3)
         echo " List of processes of users "
          ps aux
         ;;
      4)
          echo " Today's date "
          date
         ;;
      5)
         echo "Quiting program. Goodbye!"
         exit 0
         ;;
      *)
         echo "Invalid choice. Please enter a number between 1 and 5."
         ;;
   esac
done
```

Output


shobha@DESKTOP-ATHGF2M: ~
shobha@DESKTOP-ATHGF2M:~$ ./16.sh
File Handling Menu:
1. List of users who are logged in
2. List of files in the current directory
3. List of processes of user
4. Today's Date
5. Quit Unix
Enter your choice (1-5): 4
Today's date
Wed Feb 14 22:29:16 IST 2024
File Handling Menu:
1. List of users who are logged in
2. List of files in the current directory
3. List of processes of user
4. Today's Date
5. Quit Unix
Enter your choice (1-5): 3
List of processes of users
USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND
root 1 0.0 0.0 8972 116 ? Ssl 19:12 0:00 /init
root 104 0.0 0.0 9320 104 tty1 Ss 19:30 0:00 /init
shobha 105 0.0 0.0 14248 2628 tty1 S 19:30 0:00 -bash
shobha 286 0.0 0.0 12952 2124 tty1 S 22:29 0:00 /bin/bash .
shobha 288 0.0 0.0 15520 1928 tty1 R 22:29 0:00 ps aux
File Handling Menu:
1. List of users who are logged in
2. List of files in the current directory
3. List of processes of user
4. Today's Date
5. Quit Unix