Lab – 1

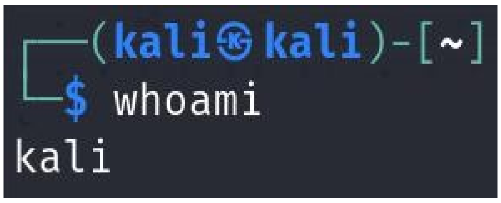
20CP207P- Operating Systems Lab

Title: Introduction of basic Linux commands.

Task: Perform the basic Linux commands and understand how it works

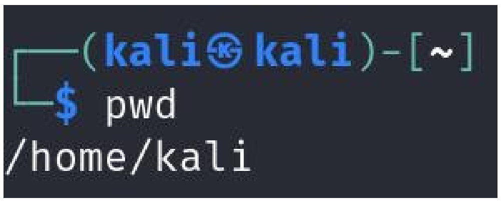
Commands:

1.whoami : Print the user name associated with the current effective user ID. Same as id -un.



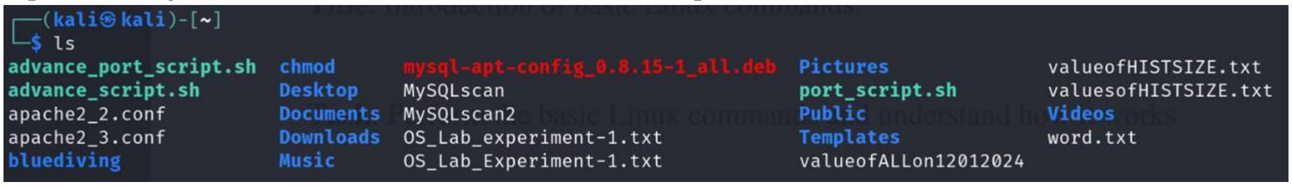
```
┌──(kali㉿kali)-[~]
└─$ whoami
kali
```

2.pwd : Print the full filename of the current working directory.



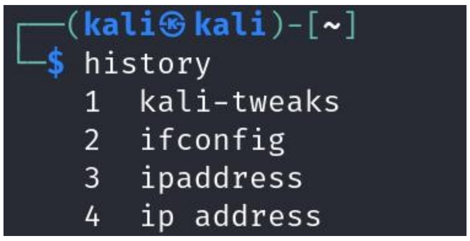
```
┌──(kali㉿kali)-[~]
└─$ pwd
/home/kali
```

3.ls : List information about the FILEs (the current directory by default). Sort entries alphabetically if none of -cf -tuvSUX nor --sort is specified.



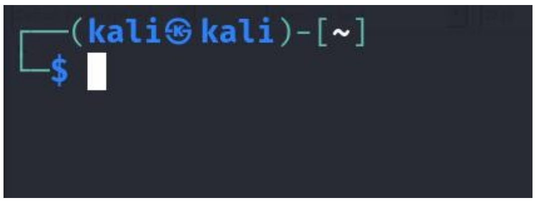
```
┌──(kali㉿kali)-[~]
└─$ ls
advance_port_script.sh  chmod      mysql-apt-config_0.8.15-1_all.deb  Pictures              valueofHISTSIZE.txt
advance_script.sh       Desktop    MySQLscan                          port_script.sh        valuesofHISTSIZE.txt
apache2_2.conf          Documents  MySQLscan2                         Public                Videos
apache2_3.conf          Downloads  OS_Lab_experiment-1.txt            Templates             word.txt
bluediving              Music      OS_Lab_Experiment-1.txt            valueofALLon12012024
```

4.history : Many programs read input from the user a line at a time. The GNU History library is able to keep track of those lines, associate arbitrary data with each line, and utilize information from previous lines in composing new ones.



```
┌──(kali㉿kali)-[~]
└─$ history
    1  kali-tweaks
    2  ifconfig
    3  ipaddress
    4  ip address
```

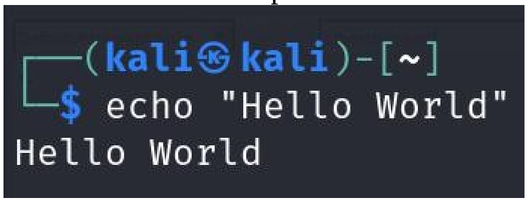
5.clear : clear clears your terminal's screen and its scrollback buffer, if any. clear retrieves the terminal type from the environment variable TERM, then consults the terminfo terminal capability database entry for that type to determine how to perform these actions.



```
┌──(kali㉿kali)-[~]
└─$ █
```

6.echo : Echo the STRING(s) to standard output.

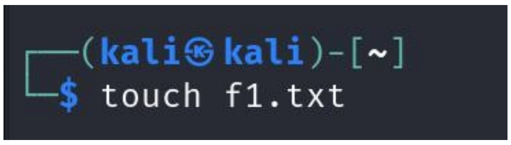
-n do not output the trailing newline
-e enable interpretation of backslash escapes
-E disable interpretation of backslash escapes (default)



```
┌──(kali㉿kali)-[~]
└─$ echo "Hello World"
Hello World
```

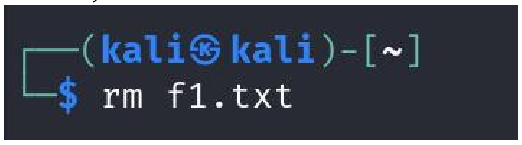
7.touch : Update the access and modification times of each FILE to the current time.

- A FILE argument that does not exist is created empty, unless -c or -h is supplied.
- A FILE argument string of - is handled specially and causes touch to change the times of the file associated with standard output.



```
┌──(kali㉿kali)-[~]
└─$ touch f1.txt
```

8.rm : This manual page documents the GNU version of rm. rm removes each specified file. By default, it does not remove directories.



```
┌──(kali㉿kali)-[~]
└─$ rm f1.txt
```

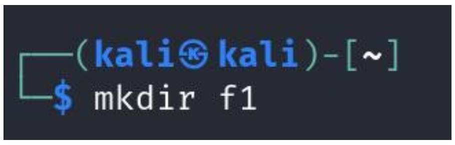
9.mkdir : Create the DIRECTORY(ies), if they do not already exist. Mandatory arguments to long options are mandatory for short options too.

-m, --mode=MODE

- set file mode (as in chmod), not a=rwx -

umask -p, --parents

- no error if existing, make parent directories as needed, with their file modes unaffected by any -m option.



```
┌──(kali㉿kali)-[~]
└─$ mkdir f1
```

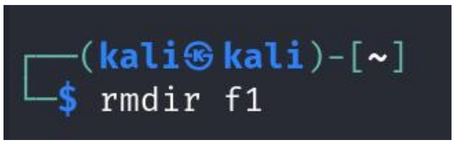
10.rmdir : Remove the DIRECTORY(ies), if they are empty.

--ignore-fail-on-non-empty

- ignore each failure that is solely because a directory is non-

empty -p, --parents

- remove DIRECTORY and its ancestors; e.g., 'rmdir -p a/b/c' is similar to 'rmdir a/b/c a/b a'



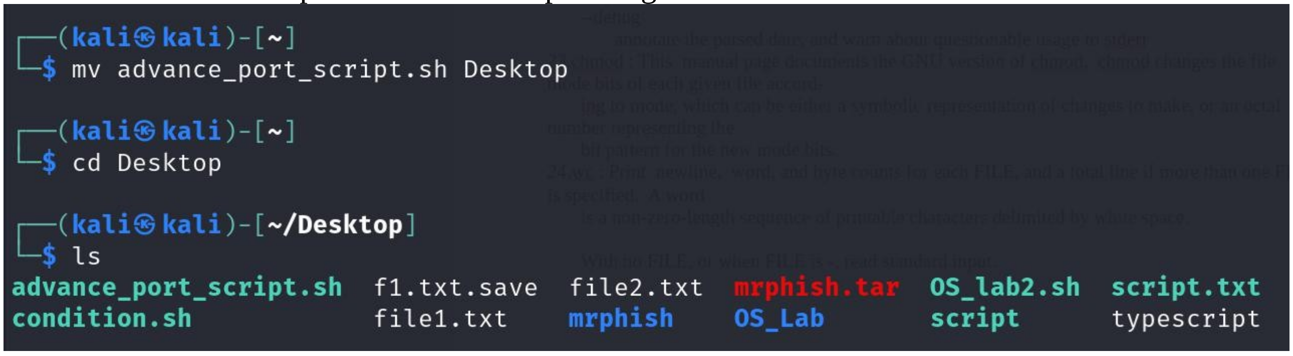
```
┌──(kali㉿kali)-[~]
└─$ rmdir f1
```

11.mv : Rename SOURCE to DEST, or move SOURCE(s) to DIRECTORY. Mandatory arguments to long options are mandatory for short options too.

--backup[=CONTROL]

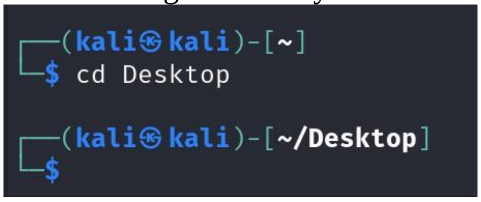
- make a backup of each existing destination

file -b like --backup but does not accept an argument



```
┌──(kali㉿kali)-[~]
└─$ mv advance_port_script.sh Desktop

┌──(kali㉿kali)-[~]
└─$ cd Desktop

┌──(kali㉿kali)-[~/Desktop]
└─$ ls
advance_port_script.sh  f1.txt.save  file2.txt  mrphish.tar  OS_lab2.sh  script.txt
condition.sh            file1.txt    mrphish    OS_Lab       script      typescript
```

12.cd : Changes directory



```
┌──(kali㉿kali)-[~]
└─$ cd Desktop

┌──(kali㉿kali)-[~/Desktop]
└─$
```

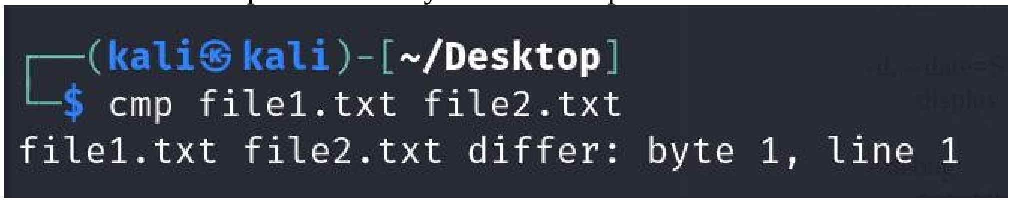
13.cmp : Compare two files byte by byte. The optional SKIP1 and SKIP2 specify the number of bytes to skip at the beginning of each file (zero by default). Mandatory arguments to long options are mandatory for short options too.

-b, --print-bytes

- print differing bytes

-i, --ignore-initial=SKIP

- skip first SKIP bytes of both inputs



```
┌──(kali㉿kali)-[~/Desktop]
└─$ cmp file1.txt file2.txt
file1.txt file2.txt differ: byte 1, line 1
```

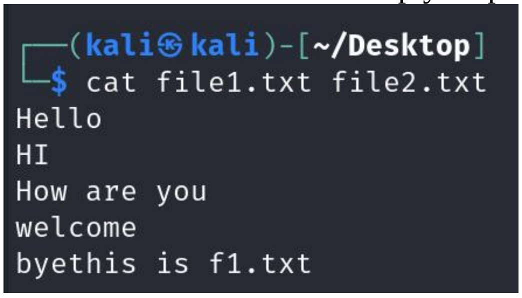
14.cat : Concatenate FILE(s) to standard output. With no FILE, or when FILE is -, read standard input.

-A, --show-all

- equivalent to -vET

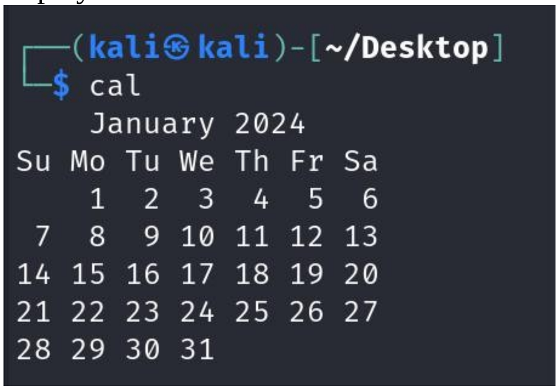
-b, --number-nonblank

- number nonempty output lines, overrides -n



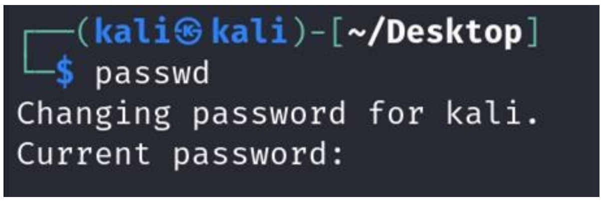
```
┌──(kali㉿kali)-[~/Desktop]
└─$ cat file1.txt file2.txt
Hello
HI
How are you
welcome
byethis is f1.txt
```

15.cal : The cal utility displays a simple calendar in traditional format and ncal offers an alternative layout, more options and the date of Easter. The new format is a little cramped but it makes a year fit on a 25x80 terminal. If arguments are not specified, the current month is displayed.



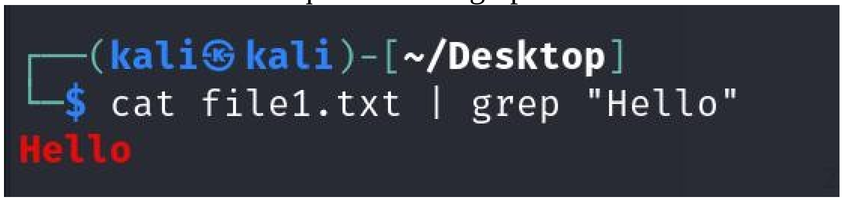
```
┌──(kali㉿kali)-[~/Desktop]
└─$ cal
    January 2024
Su Mo Tu We Th Fr Sa
    1  2  3  4  5  6
 7  8  9 10 11 12 13
14 15 16 17 18 19 20
21 22 23 24 25 26 27
28 29 30 31
```

16.passwd : The passwd command changes passwords for user accounts. A normal user may only change the password for their own account, while the superuser may change the password for any account. passwd also changes the account or associated password validity period.



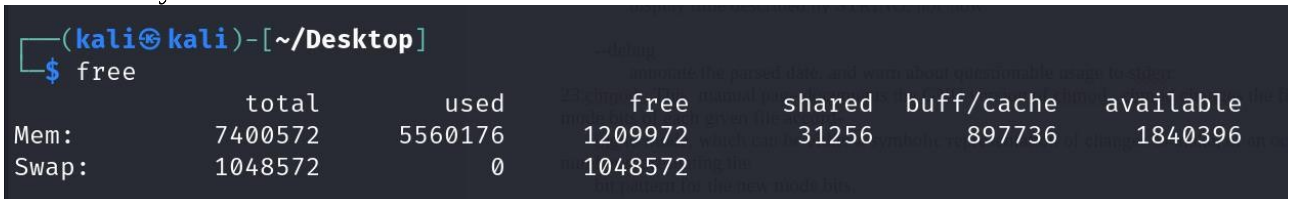
```
┌──(kali㉿kali)-[~/Desktop]
└─$ passwd
Changing password for kali.
Current password:
```

17.grep : grep searches for PATTERNS in each FILE. PATTERNS is one or more patterns separated by newline characters, and grep prints each line that matches a pattern. Typically PATTERNS should be quoted when grep is used in a shell command.



```
┌──(kali㉿kali)-[~/Desktop]
└─$ cat file1.txt | grep "Hello"
Hello
```

18.free : free displays the total amount of free and used physical and swap memory in the system, as well as the buffers and caches used by the kernel. The information is gathered by parsing /proc/meminfo. The displayed columns are: total Total usable memory (MemTotal and SwapTotal in /proc/meminfo). This includes the physical and swap memory minus a few reserved bits and kernel binary code.



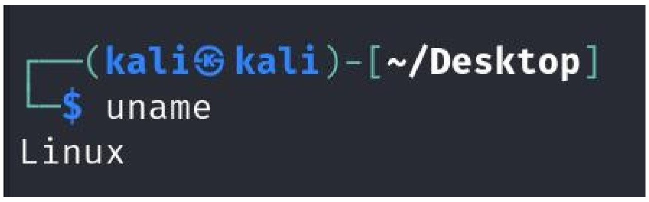
```
┌──(kali㉿kali)-[~/Desktop]
└─$ free
               total        used        free      shared  buff/cache   available
Mem:         7400572     5560176     1209972       31256      897736     1840396
Swap:        1048572           0     1048572
```

19.uname : Print certain system information. With no OPTION, same as -s.
-a, --all

- print all information, in the following order, except omit -p and -i if

unknown: -s, --kernel-name

- print the kernel name

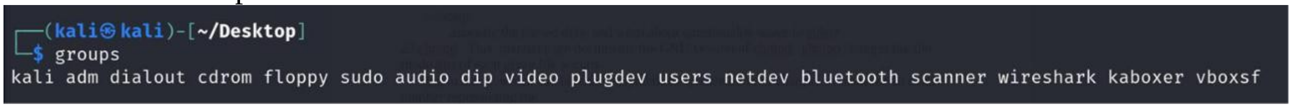


```
┌──(kali㉿kali)-[~/Desktop]
└─$ uname
Linux
```

20.groups : Print group memberships for each USERNAME or, if no USERNAME is specified, for the current process (which may differ if the groups database has changed).

--help
- display this help and exit

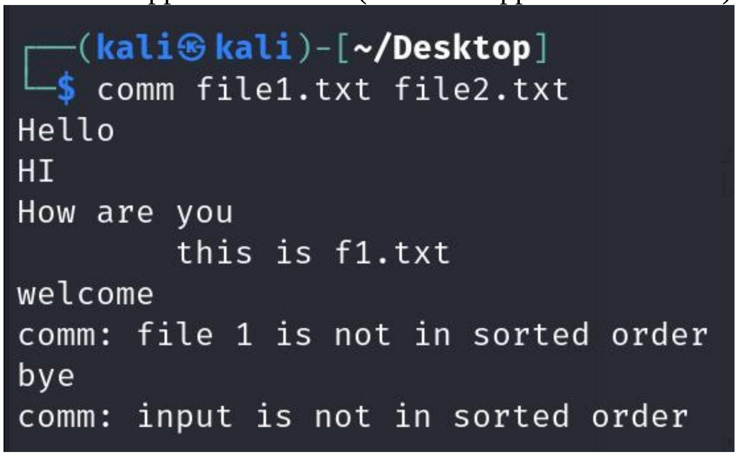
--version
- output version information and exit



```
┌──(kali㉿kali)-[~/Desktop]
└─$ groups
kali adm dialout cdrom floppy sudo audio dip video plugdev users netdev bluetooth scanner wireshark kaboxer vboxsf
```

21.comm : Compare sorted files FILE1 and FILE2 line by line. When FILE1 or FILE2 (not both) is -, read standard input. With no options, produce three-column output. Column one contains lines unique to FILE1, column two contains lines unique to FILE2, and column three contains lines common to both files.

-1 suppress column 1 (lines unique to FILE1)
-2 suppress column 2 (lines unique to FILE2)
-3 suppress column 3 (lines that appear in both files)

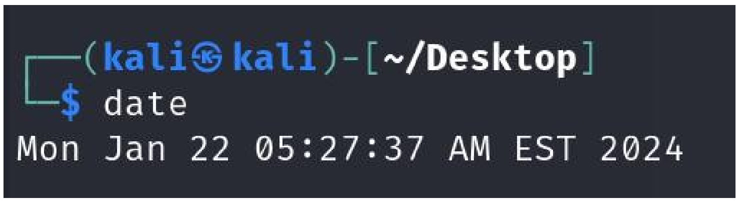


```
┌──(kali㉿kali)-[~/Desktop]
└─$ comm file1.txt file2.txt
Hello
HI
How are you
        this is f1.txt
welcome
comm: file 1 is not in sorted order
bye
comm: input is not in sorted order
```

22.Date : Display date and time in the given FORMAT. With -s, or with [MMDDhhmm[[CC]YY][.ss]], set the date and time. Mandatory arguments to long options are mandatory for short options too.

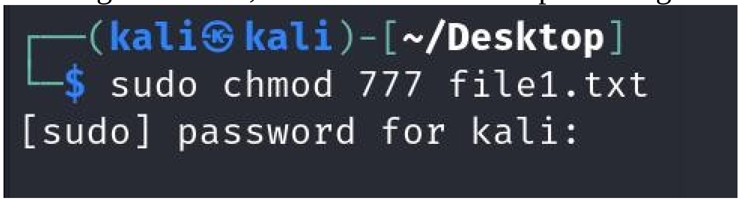
-d, --date=STRING
- display time described by STRING, not 'now'

--debug
- annotate the parsed date, and warn about questionable usage to stderr



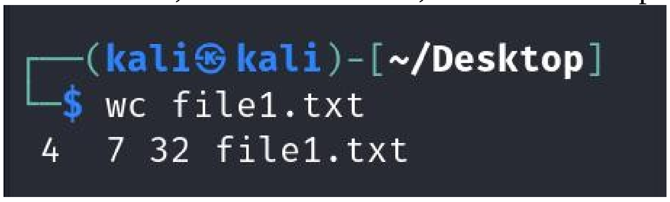
```
┌──(kali㉿kali)-[~/Desktop]
└─$ date
Mon Jan 22 05:27:37 AM EST 2024
```

23.chmod : This manual page documents the GNU version of chmod. chmod changes the file mode bits of each given file according to mode, which can be either a symbolic representation of changes to make, or an octal number representing the bit pattern for the new mode bits.



```
┌──(kali㉿kali)-[~/Desktop]
└─$ sudo chmod 777 file1.txt
[sudo] password for kali:
```

24.wc : Print newline, word, and byte counts for each FILE, and a total line if more than one FILE is specified. A word is a non-zero-length sequence of printable characters delimited by white space. With no FILE, or when FILE is -, read standard input.



```
┌──(kali㉿kali)-[~/Desktop]
└─$ wc file1.txt
 4  7 32 file1.txt
```

1 Lab – 2

1 20CP207P - Operating Systems Lab

2 **Title:** Introduction of Shell Scrip.

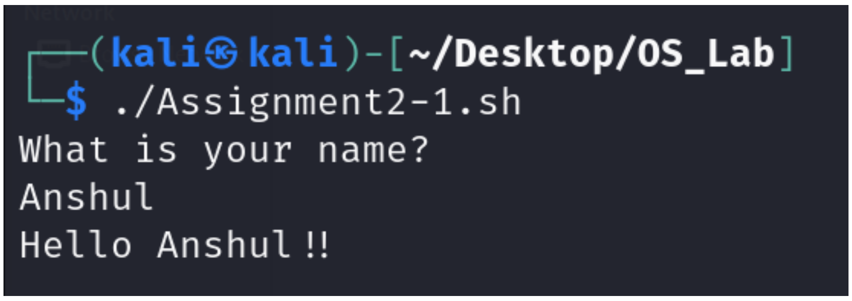
3 **Task:** Perform the basic Shell Scrip

1) Write a bash script to print from user input.

Script :

```
#! /bin/bash
echo "What is your name? "
read name
echo "Hello" $name"!!"
```

Output:



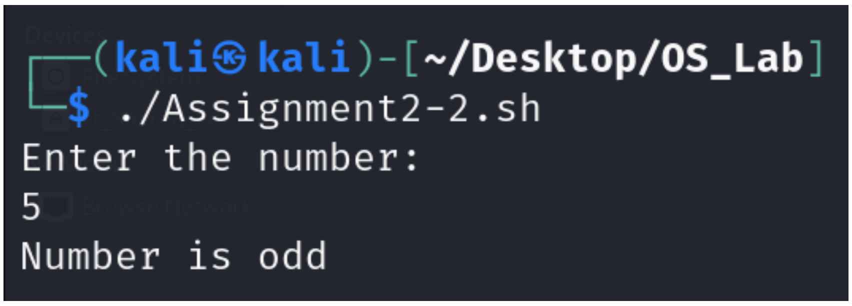
```
┌──(kali㉿kali)-[~/Desktop/OS_Lab]
└─$ ./Assignment2-1.sh
What is your name?
Anshul
Hello Anshul !!
```

2) Write a bash script to find whether a number is even or odd.

Script :

```
#! /bin/bash
echo "Enter the number: "
read num
num1=0
if [ `expr $num % 2` == $num1 ];
then
        echo "The number is even"
else
        echo "Number is odd"
fi
```

Output:



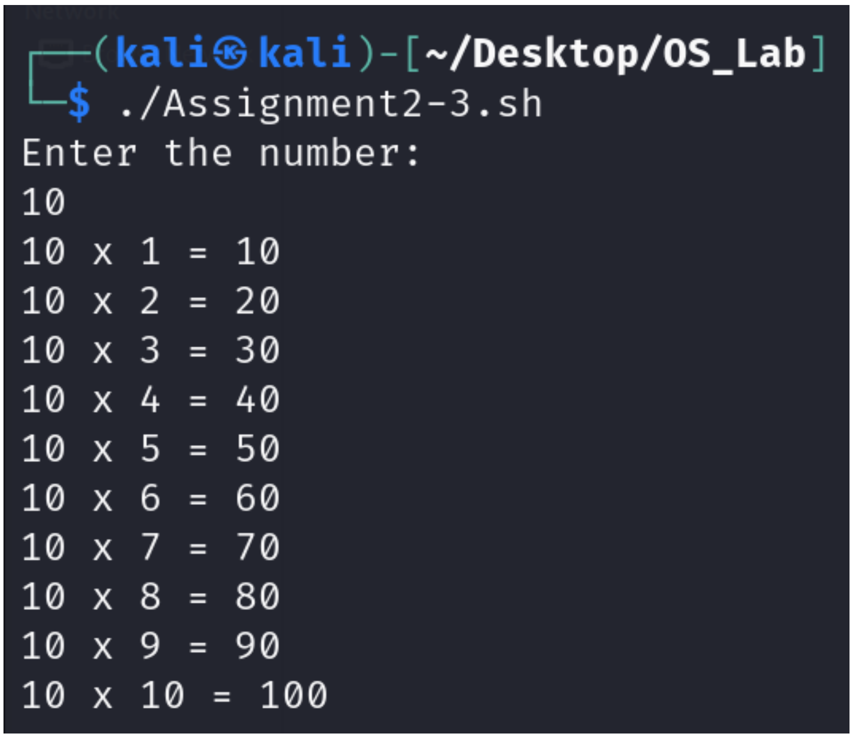
```
┌──(kali㉿kali)-[~/Desktop/OS_Lab]
└─$ ./Assignment2-2.sh
Enter the number:
5
Number is odd
```

3) Write a bash script to print the table of a given number.

Script :

```
#!/bin/bash
echo "Enter the number: "
read num
i=1
while [ $i -le 10 ]
do
        echo "$num x $i = $((num*$i))"
        let i=$((i+1))
done
```

Output:



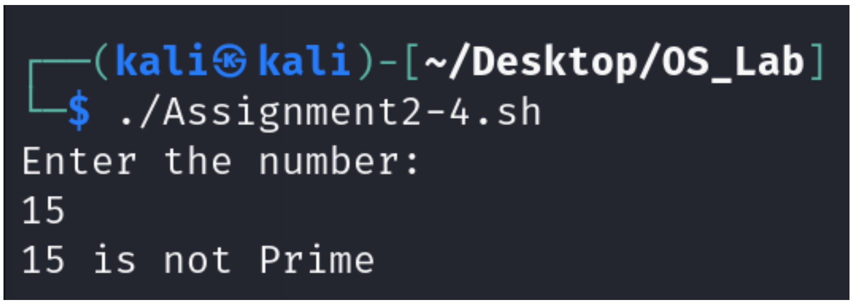
```
┌──(kali㉿kali)-[~/Desktop/OS_Lab]
└─$ ./Assignment2-3.sh
Enter the number:
10
10 x 1 = 10
10 x 2 = 20
10 x 3 = 30
10 x 4 = 40
10 x 5 = 50
10 x 6 = 60
10 x 7 = 70
10 x 8 = 80
10 x 9 = 90
10 x 10 = 100
```

4) Write a bash script to check whether a given no. is prime or not.

Script :

```
#!/bin/bash
echo "Enter the number: "
read num
i=2
j=0
num2=$((num-1))
for (( i=2;i<$num;i++ ))
do
	if [ $((num%i)) -eq 0 ]
	then
		echo "$num is not Prime"
		exit
	fi
done
echo "$num is Prime"
```

Output:



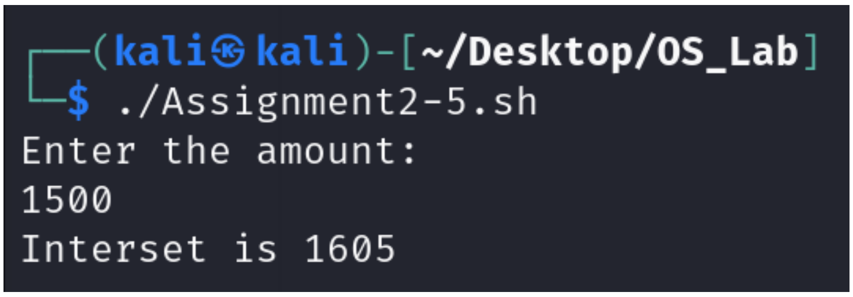
```
┌──(kali㉿kali)-[~/Desktop/OS_Lab]
└─$ ./Assignment2-4.sh
Enter the number:
15
15 is not Prime
```

5) Write a bash script to find the simple interest.

Script :

```
#!/bin/bash
echo "Enter the amount: "
read amt
interest=$(( ((amt*7)/100) + amt))
echo "Interset is $interest"
```

Output:



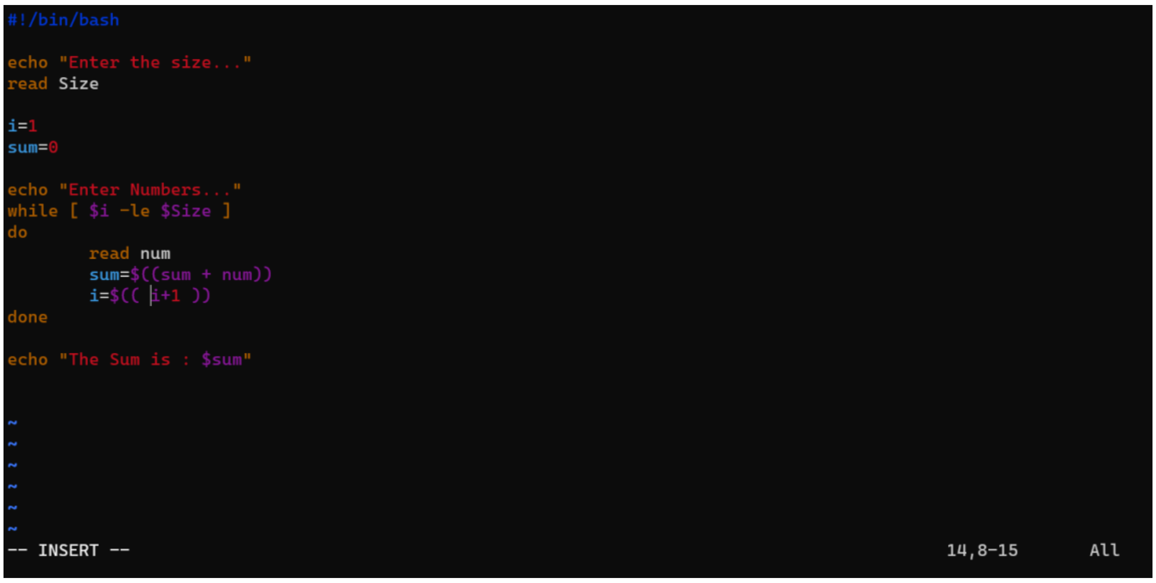
```
┌──(kali㉿kali)-[~/Desktop/OS_Lab]
└─$ ./Assignment2-5.sh
Enter the amount:
1500
Interset is 1605
```

OS Lab

Roll No:	22BCP004	Batch:	G1
Date:	02/02/24		
Practical:	3		
Title:	Introduction to shell script.		
Task:	Perform the basic Shell Scrip 1) Write a bash script to calculate the sum of n inputs. 2) Write a bash script to find the largest out of three numbers. 3) Write a menu driven bash script for the following operations. a. Display calendar of current month b. Display today's date information c. Display the username of the users currently logged in d. Display the username at given coordinates e. Display the terminal number 4) Write a bash script to get first n Fibonacci numbers. 5) Write a bash script to check whether the given year is a leap year. 6) Write a bash script to print a number triangle. 7) Write a bash script to change the input to uppercase.		

Ques:1 Write a bash script to calculate the sum of n inputs.

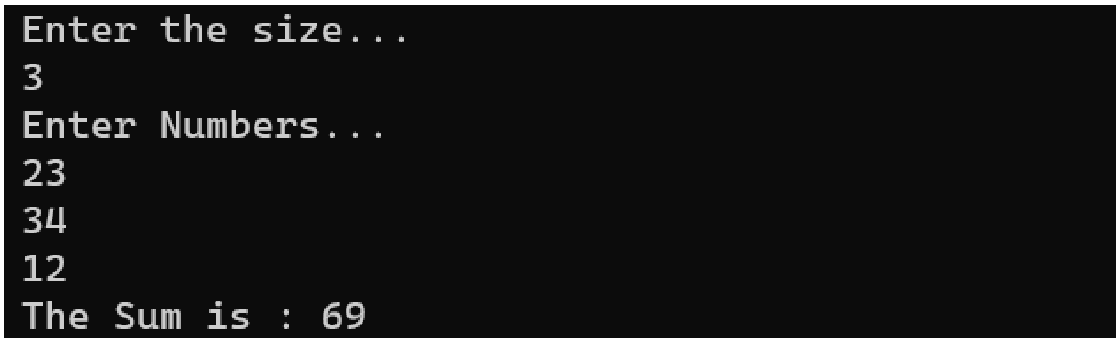
Code of Shell:



```
#!/bin/bash

echo "Enter the size..."
read Size

i=1
sum=0

echo "Enter Numbers..."
while [ $i -le $Size ]
do
        read num
        sum=$((sum + num))
        i=$(( i+1 ))
done

echo "The Sum is : $sum"


~
~
~
~
~
~
-- INSERT --                                                                                 14,8-15       All
```

Output:

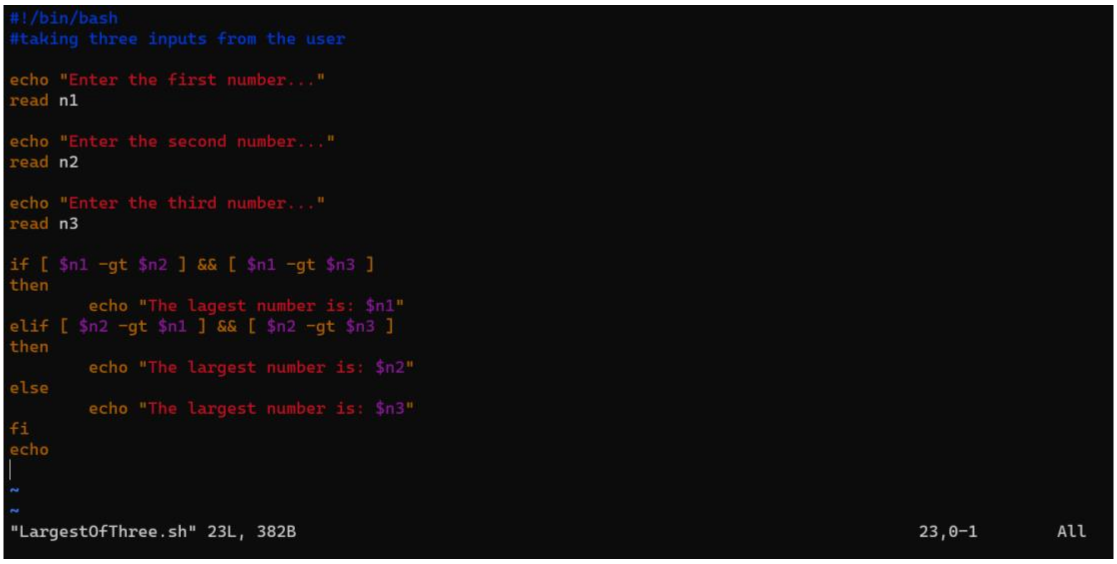
OS Lab



```
Enter the size...
3
Enter Numbers...
23
34
12
The Sum is : 69
```

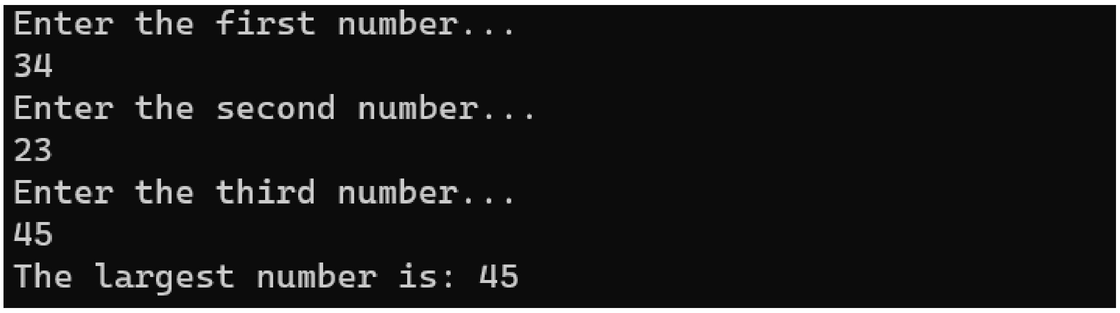
Ques: 2 Write a bash script to find the largest out of three numbers.

Code:



```
#!/bin/bash
#taking three inputs from the user

echo "Enter the first number..."
read n1

echo "Enter the second number..."
read n2

echo "Enter the third number..."
read n3

if [ $n1 -gt $n2 ] && [ $n1 -gt $n3 ]
then
        echo "The lagest number is: $n1"
elif [ $n2 -gt $n1 ] && [ $n2 -gt $n3 ]
then
        echo "The largest number is: $n2"
else
        echo "The largest number is: $n3"
fi
echo

~
~
"LargestOfThree.sh" 23L, 382B                                    23,0-1        All
```

Output:



```
Enter the first number...
34
Enter the second number...
23
Enter the third number...
45
The largest number is: 45
```

Ques: 3 Write a menu driven bash script for the following operations.

a. Display calendar of current month

b. Display today’s date information

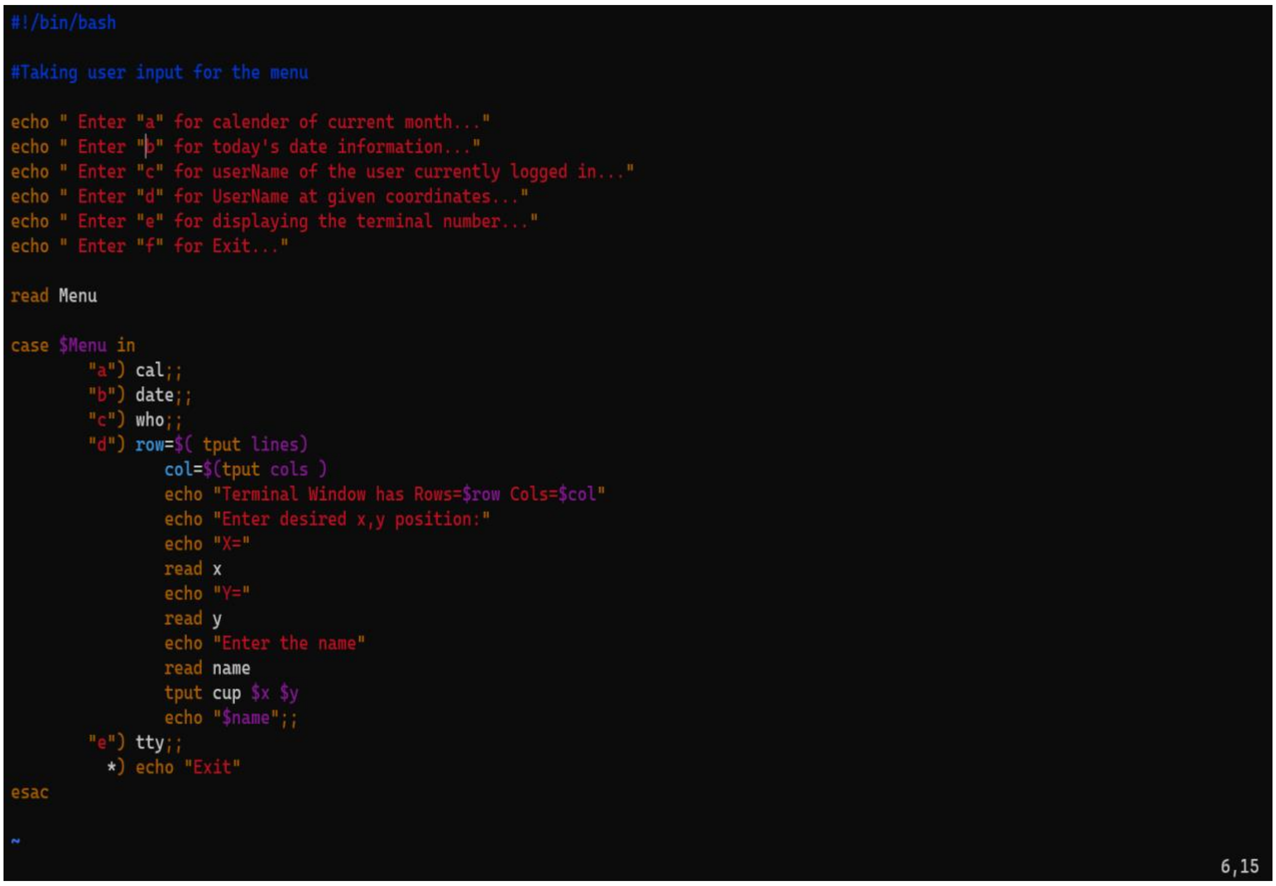
c. Display the username of the users currently logged in

OS Lab

d. Display the username at given coordinates

e. Display the terminal number

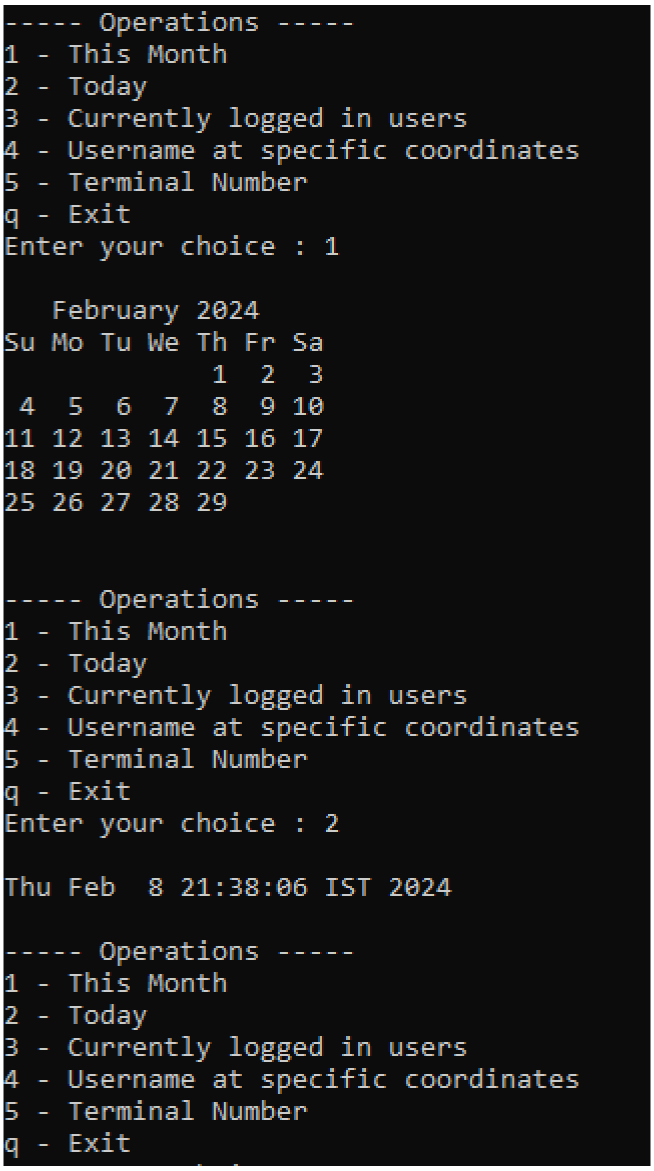
Code:



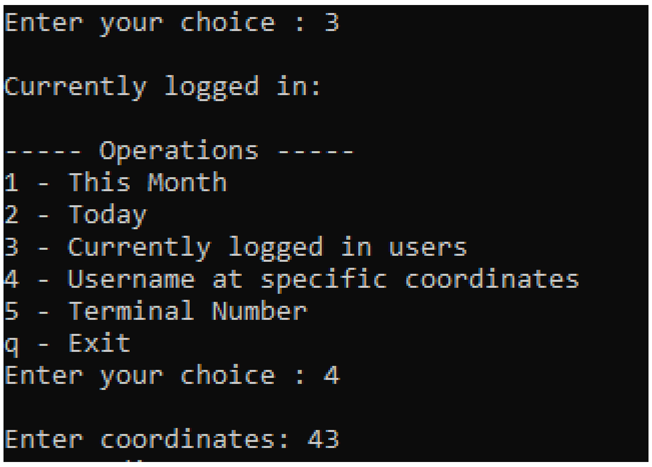
```
#!/bin/bash

#Taking user input for the menu

echo " Enter "a" for calender of current month..."
echo " Enter "b" for today's date information..."
echo " Enter "c" for userName of the user currently logged in..."
echo " Enter "d" for UserName at given coordinates..."
echo " Enter "e" for displaying the terminal number..."
echo " Enter "f" for Exit..."

read Menu

case $Menu in
        "a") cal;;
        "b") date;;
        "c") who;;
        "d") row=$( tput lines)
                col=$(tput cols )
                echo "Terminal Window has Rows=$row Cols=$col"
                echo "Enter desired x,y position:"
                echo "X="
                read x
                echo "Y="
                read y
                echo "Enter the name"
                read name
                tput cup $x $y
                echo "$name";;
        "e") tty;;
          *) echo "Exit"
esac

~
                                                                      6,15
```

Output:

OS Lab

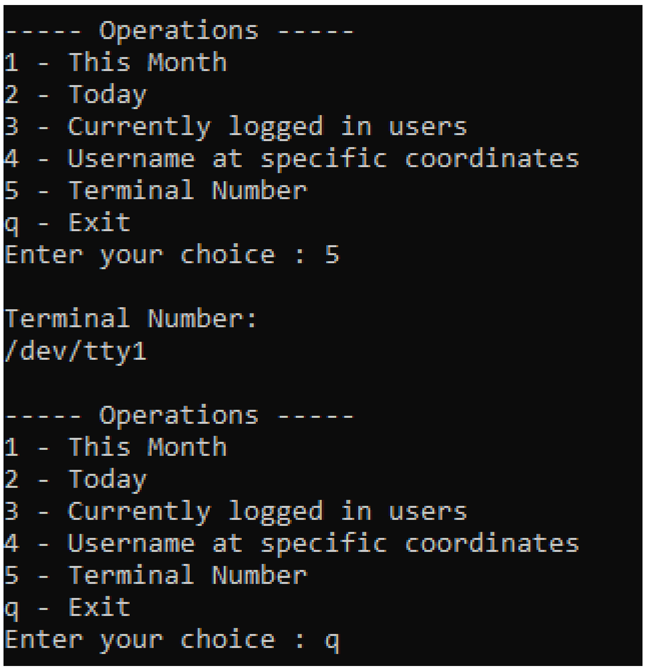


```
----- Operations -----
1 - This Month
2 - Today
3 - Currently logged in users
4 - Username at specific coordinates
5 - Terminal Number
q - Exit
Enter your choice : 1

   February 2024
Su Mo Tu We Th Fr Sa
             1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29


----- Operations -----
1 - This Month
2 - Today
3 - Currently logged in users
4 - Username at specific coordinates
5 - Terminal Number
q - Exit
Enter your choice : 2

Thu Feb  8 21:38:06 IST 2024

----- Operations -----
1 - This Month
2 - Today
3 - Currently logged in users
4 - Username at specific coordinates
5 - Terminal Number
q - Exit
```



```
Enter your choice : 3

Currently logged in:

----- Operations -----
1 - This Month
2 - Today
3 - Currently logged in users
4 - Username at specific coordinates
5 - Terminal Number
q - Exit
Enter your choice : 4

Enter coordinates: 43
```

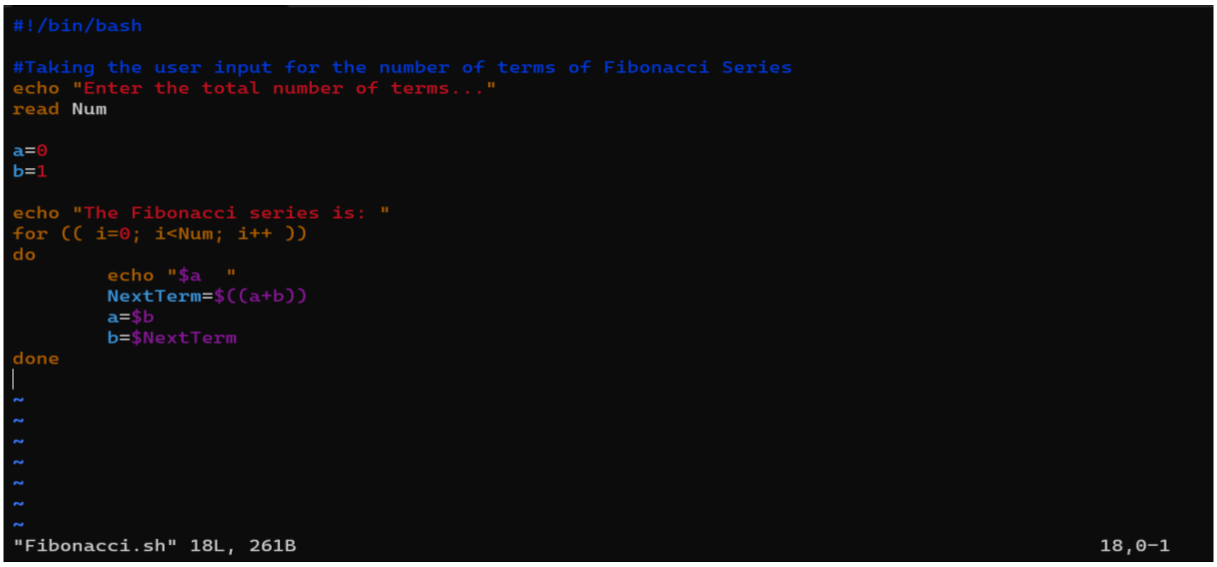
OS Lab



```
----- Operations -----
1 - This Month
2 - Today
3 - Currently logged in users
4 - Username at specific coordinates
5 - Terminal Number
q - Exit
Enter your choice : 5

Terminal Number:
/dev/tty1

----- Operations -----
1 - This Month
2 - Today
3 - Currently logged in users
4 - Username at specific coordinates
5 - Terminal Number
q - Exit
Enter your choice : q
```

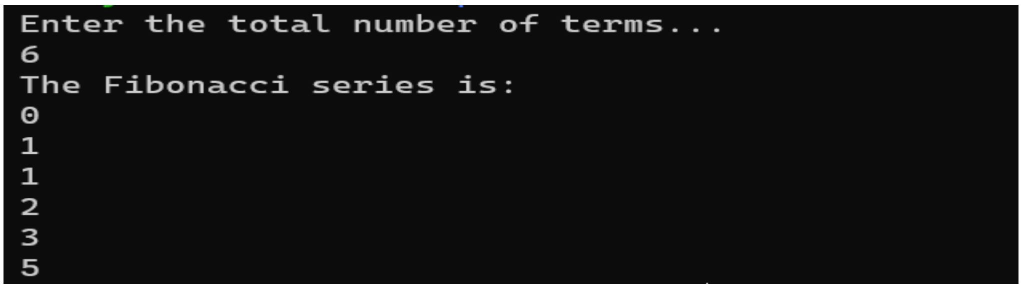
Ques: 4 Write a bash script to get first n Fibonacci numbers

Code:



```
#!/bin/bash

#Taking the user input for the number of terms of Fibonacci Series
echo "Enter the total number of terms..."
read Num

a=0
b=1

echo "The Fibonacci series is: "
for (( i=0; i<Num; i++ ))
do
        echo "$a  "
        NextTerm=$((a+b))
        a=$b
        b=$NextTerm
done

~
~
~
~
~
~
~
"Fibonacci.sh" 18L, 261B                                                    18,0-1
```

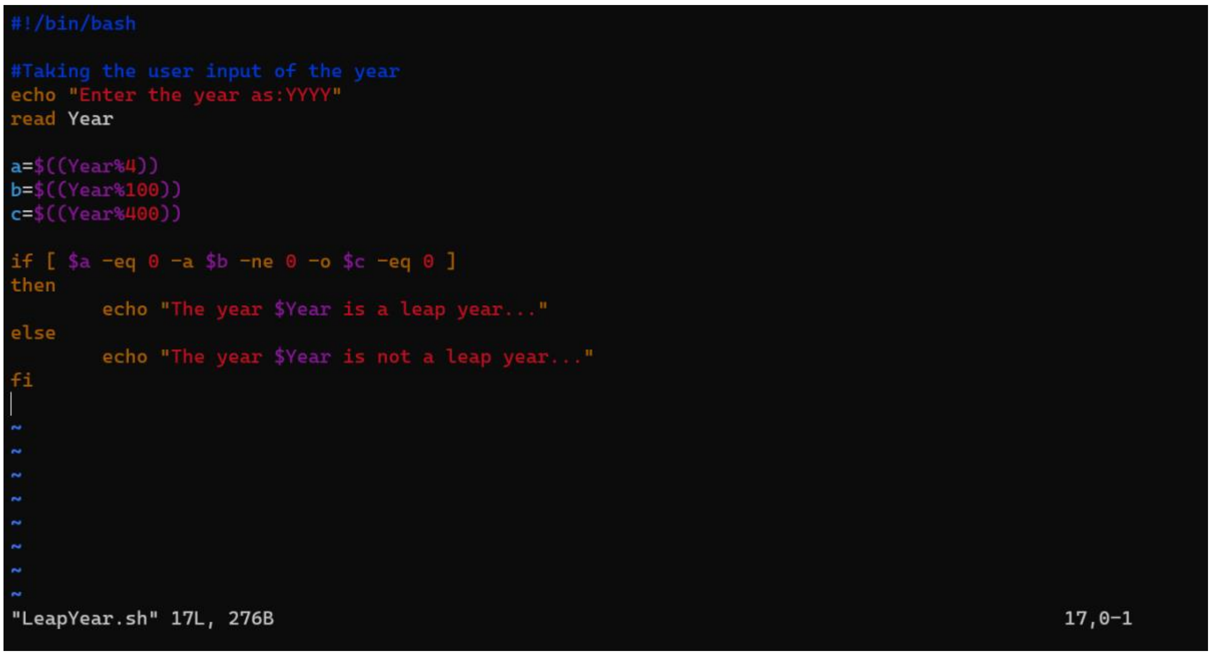
Output:



```
Enter the total number of terms...
6
The Fibonacci series is:
0
1
1
2
3
5
```

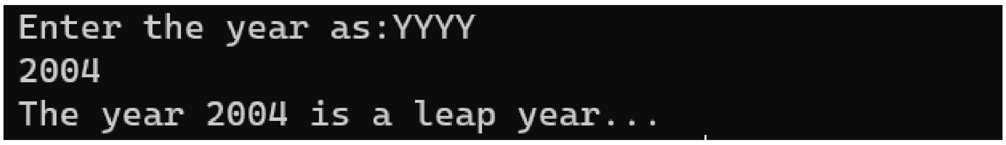
Ques: 5 Write a bash script to check whether the given year is a leap year.

Code:



```
#!/bin/bash

#Taking the user input of the year
echo "Enter the year as:YYYY"
read Year

a=$((Year%4))
b=$((Year%100))
c=$((Year%400))

if [ $a -eq 0 -a $b -ne 0 -o $c -eq 0 ]
then
        echo "The year $Year is a leap year..."
else
        echo "The year $Year is not a leap year..."
fi

~
~
~
~
~
~
~
~
"LeapYear.sh" 17L, 276B                                              17,0-1
```

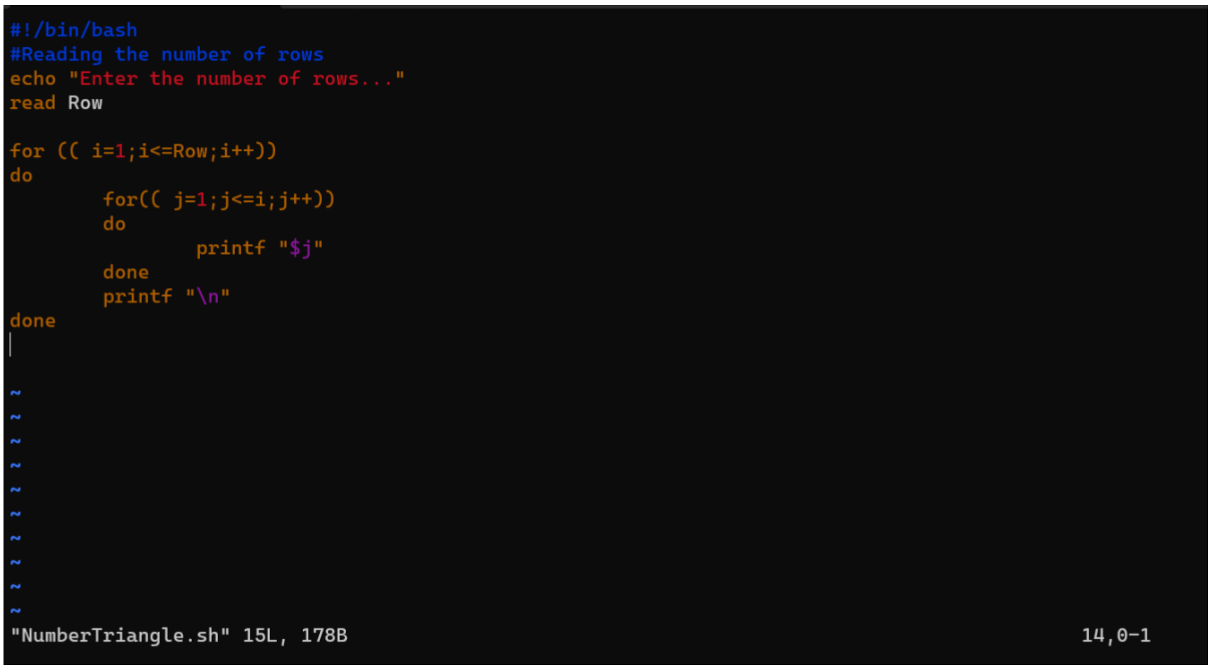
Output:



```
Enter the year as:YYYY
2004
The year 2004 is a leap year...
```

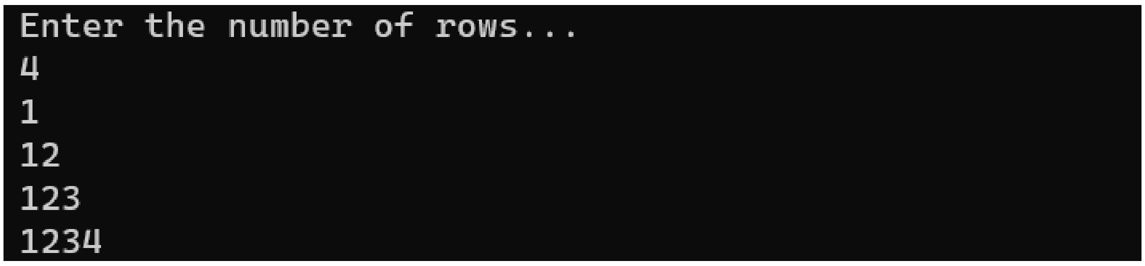
Ques: 6 Write a bash script to print a number triangle.

Code:



```
#!/bin/bash
#Reading the number of rows
echo "Enter the number of rows..."
read Row

for (( i=1;i<=Row;i++))
do
        for(( j=1;j<=i;j++))
        do
                printf "$j"
        done
        printf "\n"
done

~
~
~
~
~
~
~
~
~
~
"NumberTriangle.sh" 15L, 178B                                                              14,0-1
```

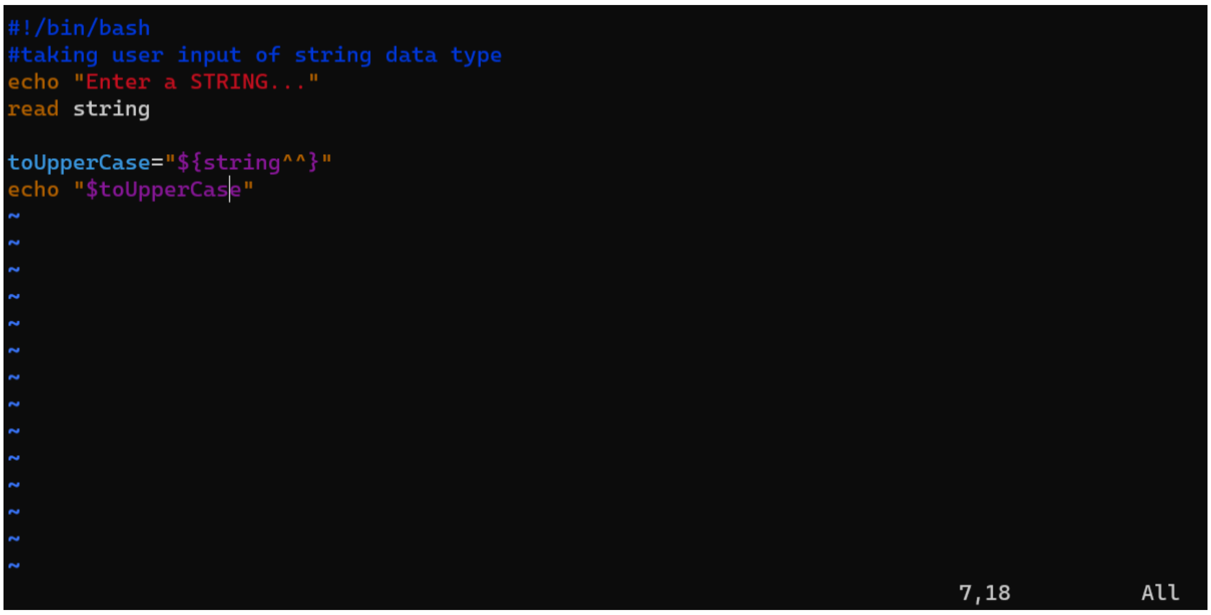
Output:



```
Enter the number of rows...
4
1
12
123
1234
```

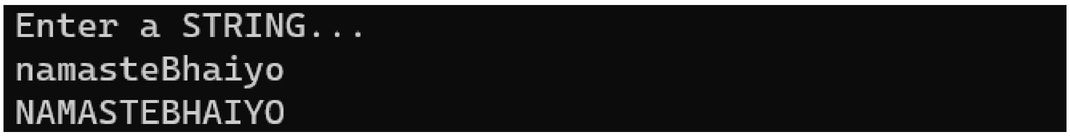
Ques: 7 Write a bash script to change the input to uppercase.

Code:



```
#!/bin/bash
#taking user input of string data type
echo "Enter a STRING..."
read string

toUpperCase="${string^^}"
echo "$toUpperCase"
~
~
~
~
~
~
~
~
~
~
~
~
~
~
                                                                  7,18          All
```

Output:



```
Enter a STRING...
namasteBhaiyo
NAMASTEBHAIYO
```

OPERATING SYSTEM

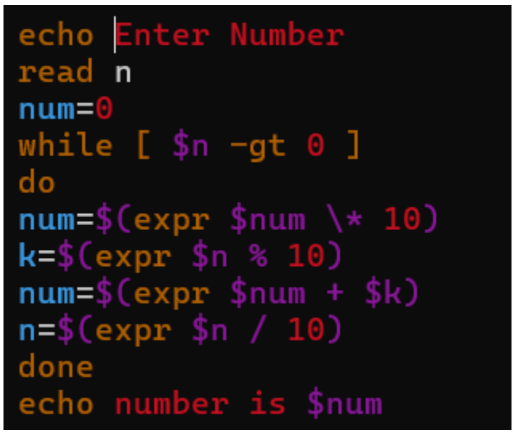
Lab Assignment - 04

Name: Meet Dholakia

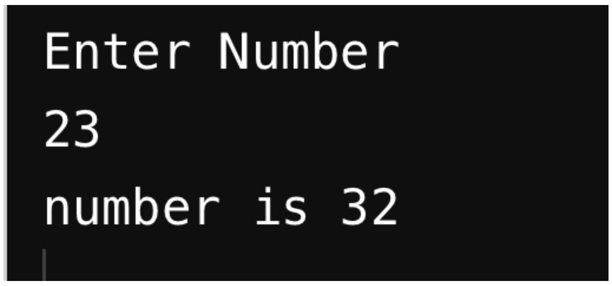
Roll no.: 22BCP004

Q1. Write a bash script to print the reverse of a given number.

Ans.=>



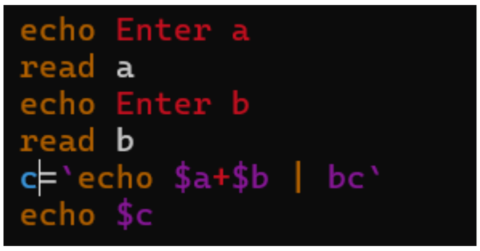
```
echo Enter Number
read n
num=0
while [ $n -gt 0 ]
do
num=$(expr $num \* 10)
k=$(expr $n % 10)
num=$(expr $num + $k)
n=$(expr $n / 10)
done
echo number is $num
```



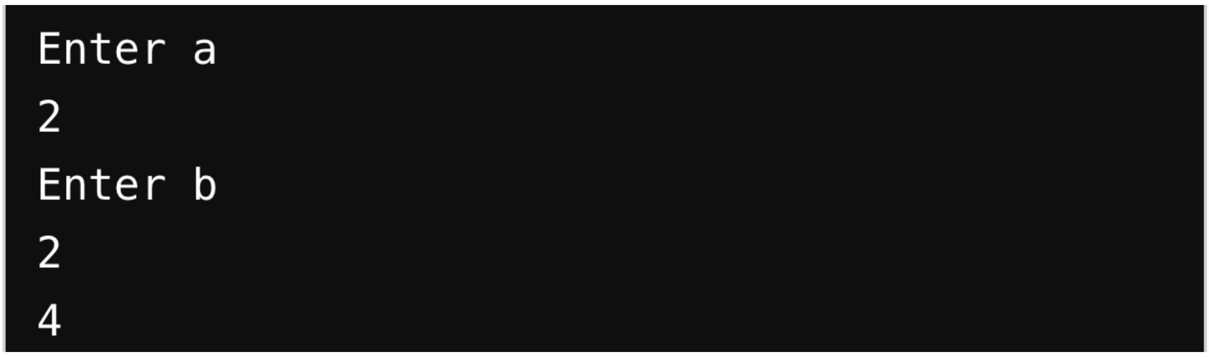
```
Enter Number
23
number is 32
```

Q2. Write a bash script to add 2 float numbers.

Ans.=>



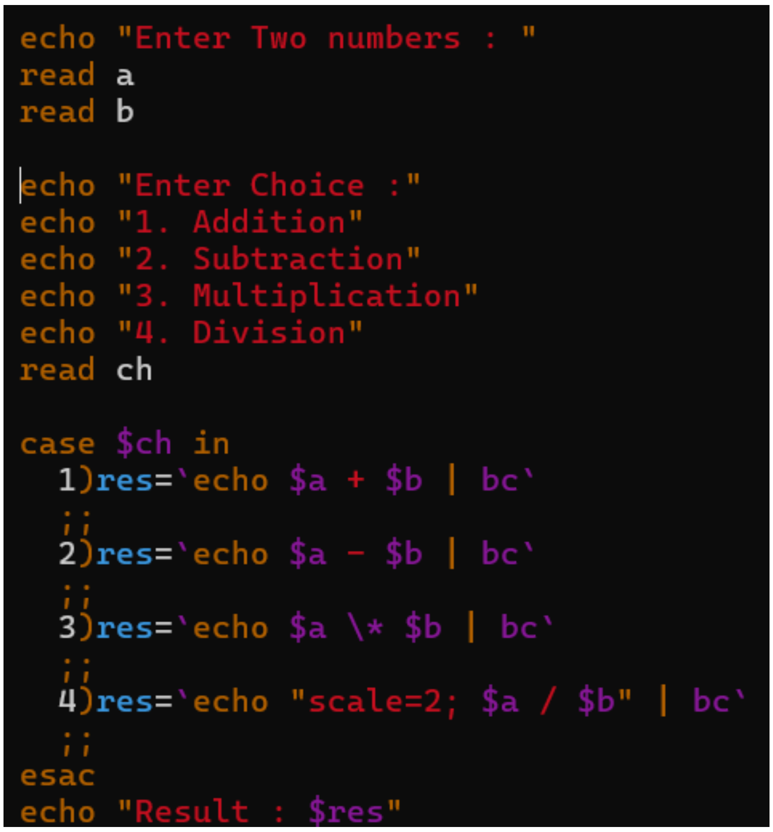
```
echo Enter a
read a
echo Enter b
read b
c=`echo $a+$b | bc`
echo $c
```



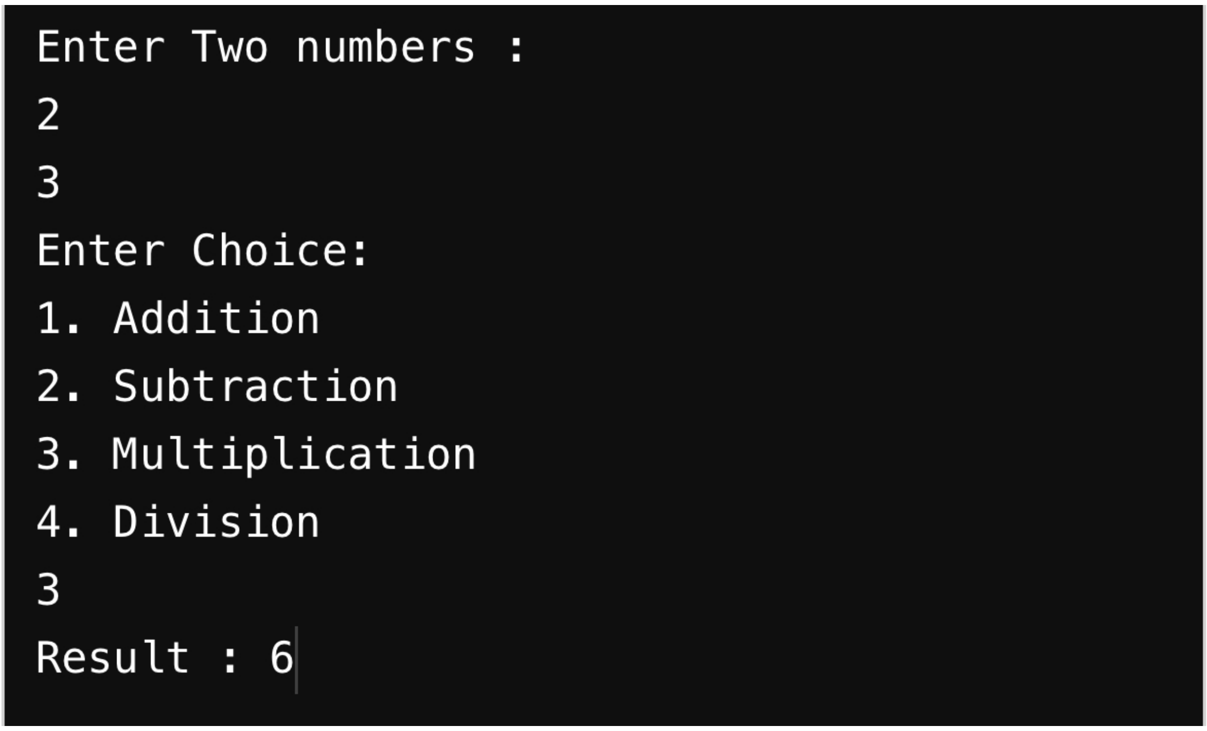
```
Enter a
2
Enter b
2
4
```

Q3. Write a bash script for a menu driven calculator for + -* /

Ans.=>



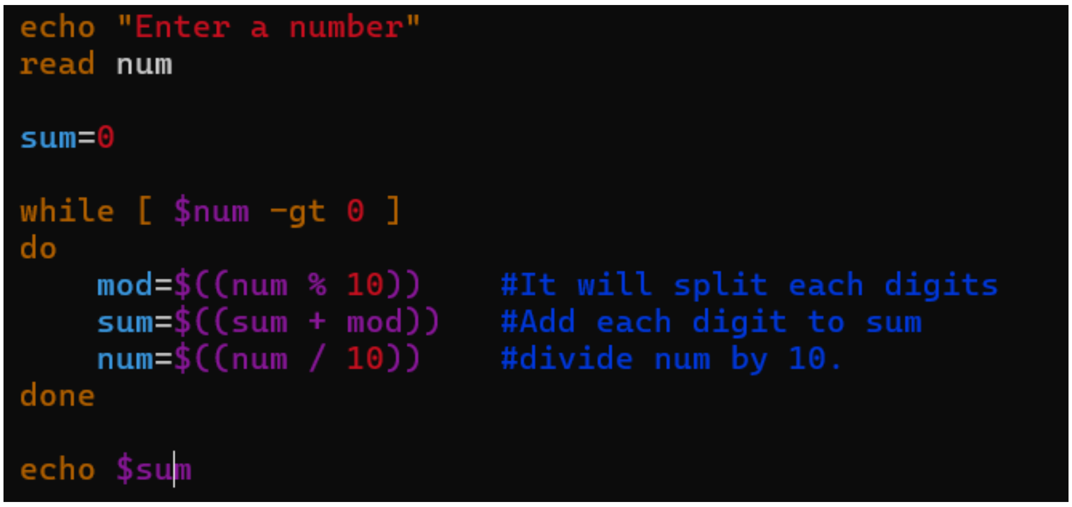
```
echo "Enter Two numbers : "
read a
read b

echo "Enter Choice :"
echo "1. Addition"
echo "2. Subtraction"
echo "3. Multiplication"
echo "4. Division"
read ch

case $ch in
  1)res=`echo $a + $b | bc`
  ;;
  2)res=`echo $a - $b | bc`
  ;;
  3)res=`echo $a \* $b | bc`
  ;;
  4)res=`echo "scale=2; $a / $b" | bc`
  ;;
esac
echo "Result : $res"
```



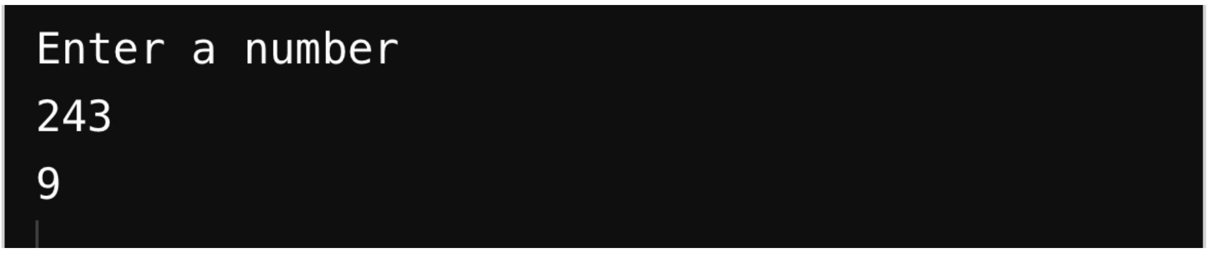
```
Enter Two numbers :
2
3
Enter Choice:
1. Addition
2. Subtraction
3. Multiplication
4. Division
3
Result : 6
```

Q4. Write a bash script to add the digits of a number.

Ans.=>



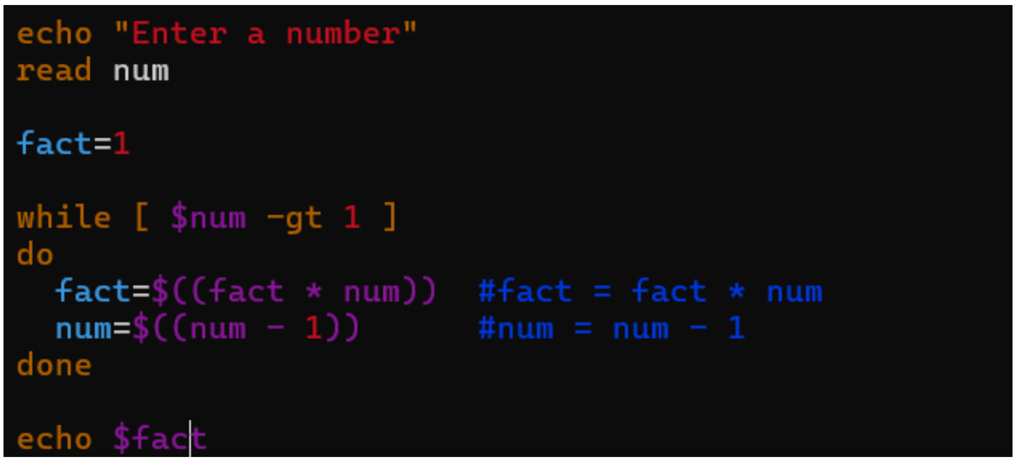
```
echo "Enter a number"
read num

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
```

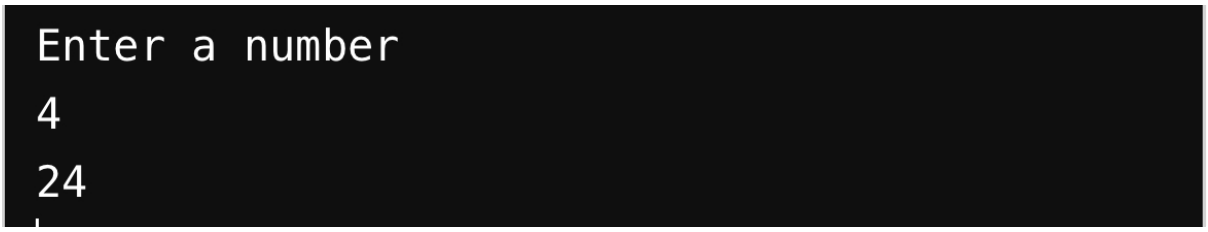


```
Enter a number
243
9
```

Q5. Write a bash script to print the factorial of a number.

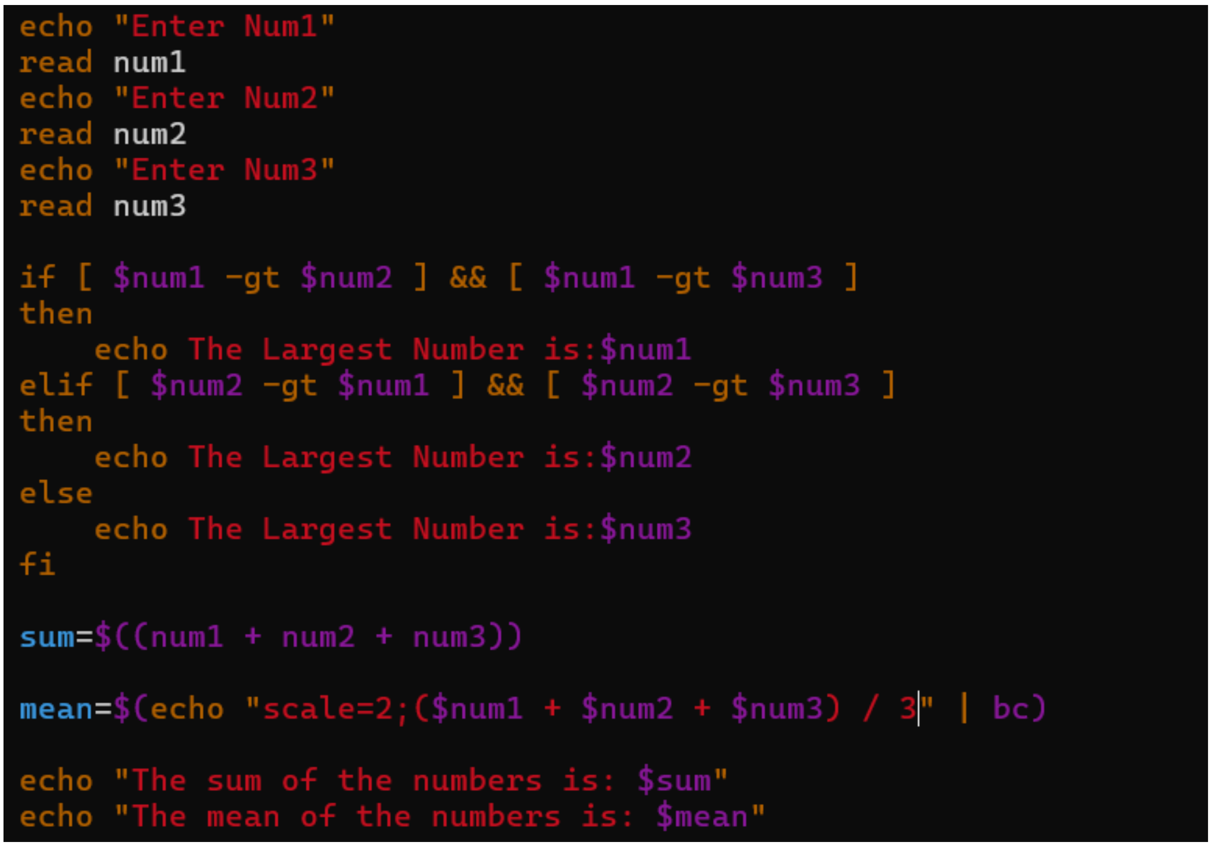
Ans.=>

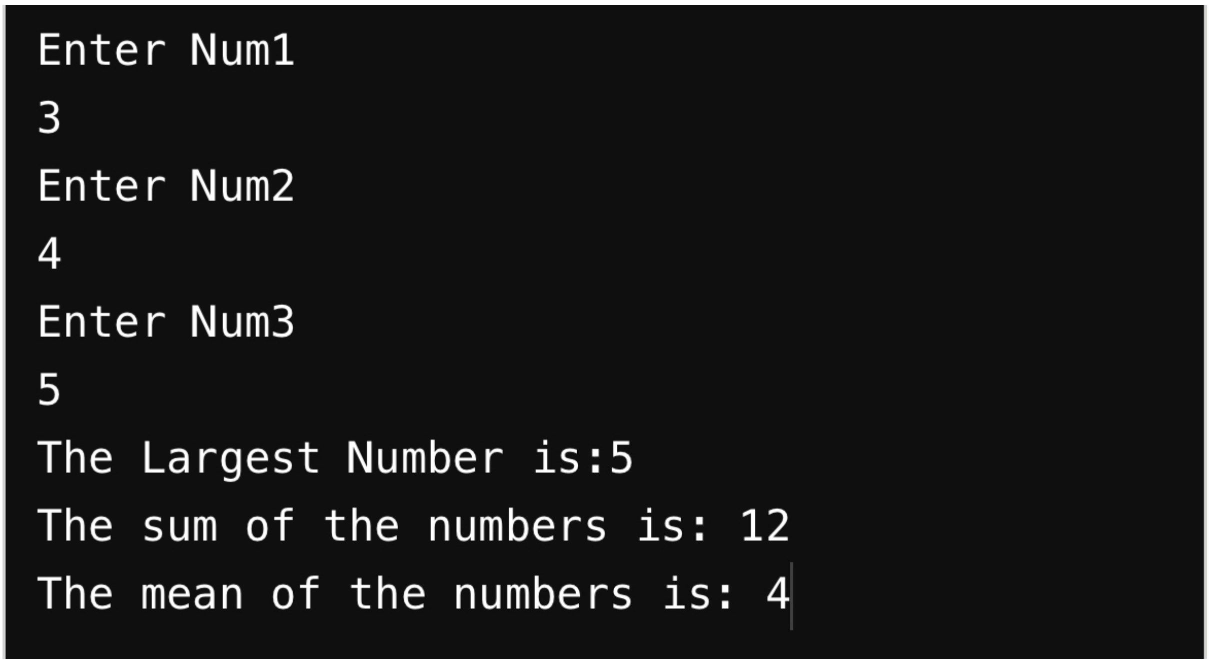

```
echo "Enter a number"
read num

fact=1

while [ $num -gt 1 ]
do
   fact=$((fact * num))  #fact = fact * num
   num=$((num - 1))      #num = num - 1
done

echo $fact
```


```
Enter a number
4
24
```

Q6. Write a shell script to find the largest of three numbers and also find the sum and the mean.

Ans.=>

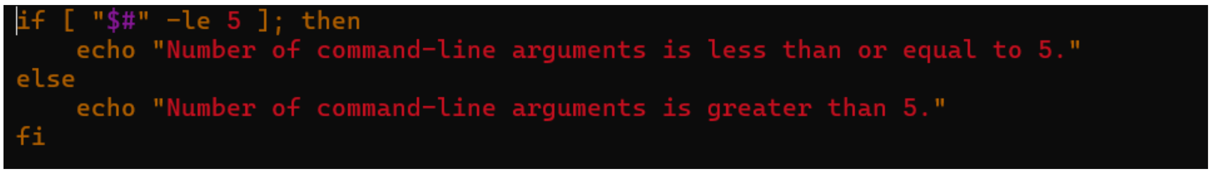

```
echo "Enter Num1"
read num1
echo "Enter Num2"
read num2
echo "Enter Num3"
read num3

if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]
then
    echo The Largest Number is:$num1
elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]
then
    echo The Largest Number is:$num2
else
    echo The Largest Number is:$num3
fi

sum=$((num1 + num2 + num3))

mean=$(echo "scale=2;($num1 + $num2 + $num3) / 3" | bc)

echo "The sum of the numbers is: $sum"
echo "The mean of the numbers is: $mean"
```



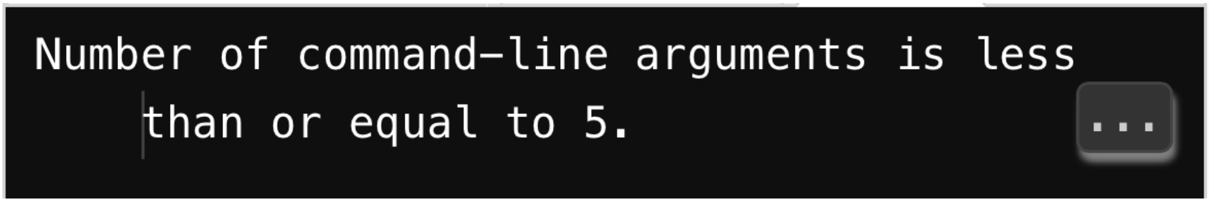
```
Enter Num1
3
Enter Num2
4
Enter Num3
5
The Largest Number is:5
The sum of the numbers is: 12
The mean of the numbers is: 4
```

Q7. Write a shell script to check whether the number of command line arguments passed are less than or equal to 5.

Ans.=>



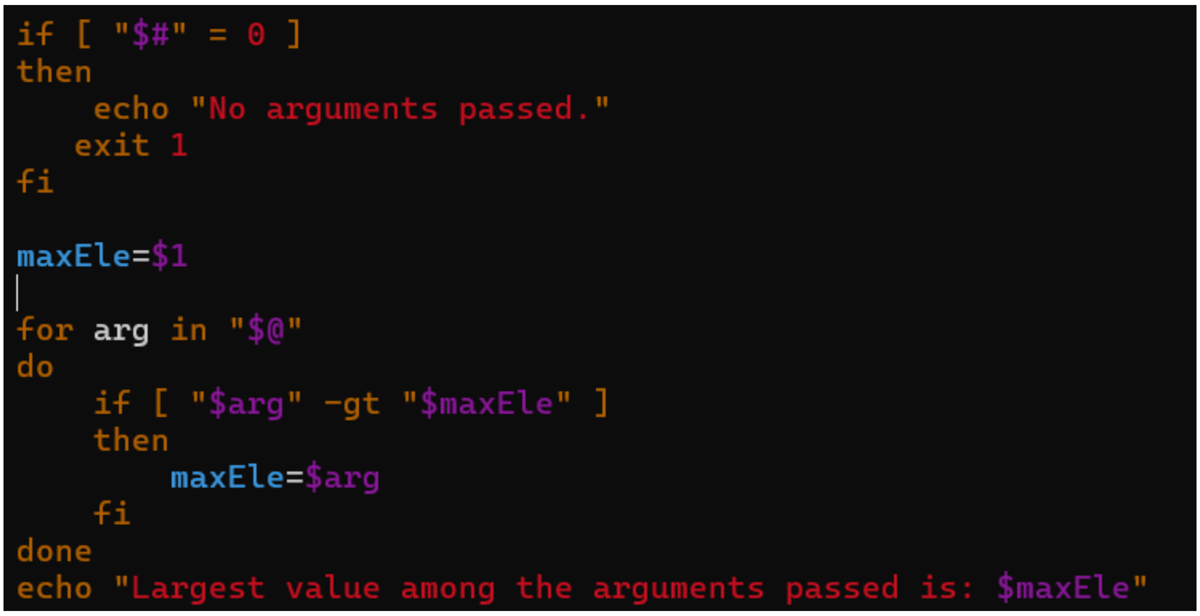
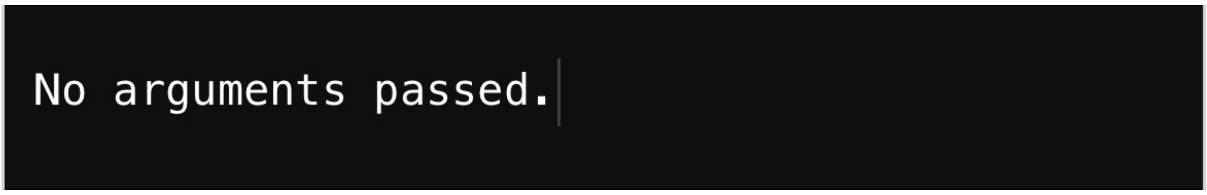
```
if [ "$#" -le 5 ]; then
    echo "Number of command-line arguments is less than or equal to 5."
else
    echo "Number of command-line arguments is greater than 5."
fi
```



```
Number of command-line arguments is less
     than or equal to 5.
```

Q8. Write a bash script to print the maximum from command line arguments.

Ans.=>


```
if [ "$#" = 0 ]
then
    echo "No arguments passed."
   exit 1
fi

maxEle=$1

for arg in "$@"
do
    if [ "$arg" -gt "$maxEle" ]
    then
        maxEle=$arg
    fi
done
echo "Largest value among the arguments passed is: $maxEle"
```

```
No arguments passed.
```