

# Operating System

---

## PRACTICAL LIST

Submitted By:

Ankit

19MTS5630

BSc. Mathematical Science

Submitted to:

Dr. Shweta Wadhera

# Index

SNO.	PROGRAM NAME	PAGE NO.
1.	Write a shell script to check if the number entered at the command line is prime or not.	1
2.	Write a shell script to modify "cal" command to display calendars of the specified months.	2
3.	Write a shell script to modify "cal" command to display calendars of the specified range of months.	3
4.	Write a shell script to accept a login name. If not a valid login name display message - "Entered login name is invalid".	5
5.	Write a shell script to display date in the mm/dd/yy format.	6
6.	Write a shell script to display on the screen sorted output of "who" command along with the total number of users.	7
7.	Write a shell script to display the multiplication table any number.	8
8.	Write a shell script to compare two files and if found equal asks the user to delete the duplicate file.	9
9.	Write a shell script to find the sum of digits of a given number.	11
10.	Write a shell script to merge the contents of three files, sort the contents and then display them page by page.	12
11.	Write a shell script to find the LCD (least common divisor) of two numbers.	14
12.	Write a shell script to perform the tasks of basic calculator.	15
13.	Write a shell script to find the power of a given number.	18
14.	Write a shell script to find the factorial of a given number.	19
15.	Write a shell script to check whether the number is Armstrong or not.	20
16.	Write a shell script to check whether the file have all the permissions or not.	21
17.	Write a shell script to show the "Pyramid" of special character "*".	23

**Q1. Write a shell script to check if the number entered at the command line is prime or not.**

**Code:**

```
echo "-----"
echo "Program to find out the given number is prime or not."
echo ""

echo -n "Enter a number: "
read n

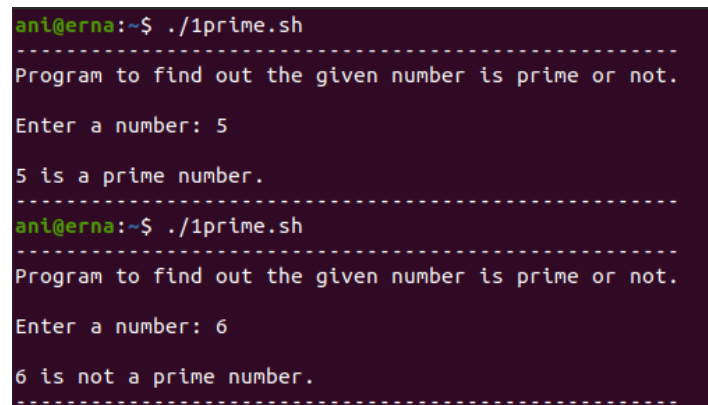
i=2
f=0

while [ $i -le `expr $n / 2` ]
do
    if [ `expr $n % $i` -eq 0 ]
    then
        f=1
    fi
    i=`expr $i + 1`
done

echo ""

if [ $f -eq 1 ]
then
    echo "$n is not a prime number."
else
    echo "$n is prime number."
fi
echo "-----"
```

**Output:**



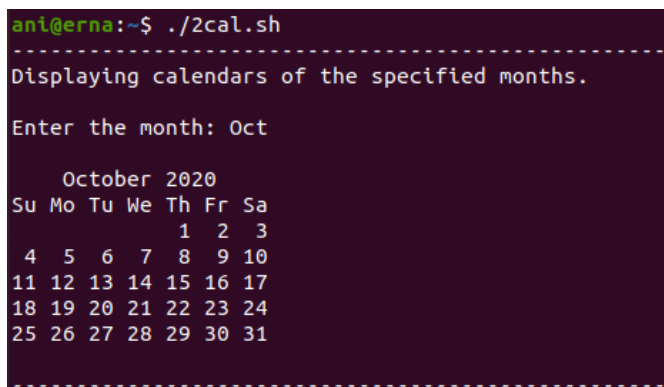
```
ani@erna:~$ ./1prime.sh
-----
Program to find out the given number is prime or not.
Enter a number: 5
5 is a prime number.
-----
ani@erna:~$ ./1prime.sh
-----
Program to find out the given number is prime or not.
Enter a number: 6
6 is not a prime number.
-----
```

**Q2. Write a shell script to modify “cal” command to display calendars of the specified months.**

**Code:**

```
echo "-----"
echo "Displaying calendars of the specified months."
echo ""
echo -n "Enter the month: " ; read ans
for month in $ans
do
    m=`echo $month | tr '[A-Z]' '[a-z]`
    case $m in
        jan*) m=1 ;;
        feb*) m=2 ;;
        mar*) m=3 ;;
        apr*) m=4 ;;
        may*) m=5 ;;
        jun*) m=6 ;;
        jul*) m=7 ;;
        aug*) m=8 ;;
        sep*) m=9 ;;
        oct*) m=10 ;;
        nov*) m=11 ;;
        dec*) m=12 ;;
        *) m="fail"
    esac
    set `date` ; y=$4
    echo ""
    if [ $m == "fail" ]
    then
        echo "Invalid Input!!"
    else
        cal $m $y
    fi
done
echo "-----"
```

**Output:**



```
ani@erna:~$ ./2cal.sh
-----
Displaying calendars of the specified months.
Enter the month: Oct

    October 2020
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
-----
```

**Q3. Write a shell script to modify “cal” command to display calendars of the specified range of months.**

**Code:**

```
echo "-----"
echo "Program to modify cal command to display calendars of the specified range of months."
echo ""

echo -n "Enter first month of range: "
read first
first=`echo $first | tr -s "." | cut -d"." -f1`

echo -n "Enter last month of range: "
read last
last=`echo $last | tr -s "." | cut -d"." -f2`

month1=`echo $first | tr '[A-Z]' '[a-z]`

case $month1 in
    jan*) m=1 ;;
    feb*) m=2 ;;
    mar*) m=3 ;;
    apr*) m=4 ;;
    may*) m=5 ;;
    jun*) m=6 ;;
    jul*) m=7 ;;
    aug*) m=8 ;;
    sep*) m=9 ;;
    oct*) m=10 ;;
    nov*) m=11 ;;
    dec*) m=12 ;;
esac

month2=`echo $last | tr '[A-Z]' '[a-z]`

case $month2 in
    jan*) n=1 ;;
    feb*) n=2 ;;
    mar*) n=3 ;;
    apr*) n=4 ;;
    may*) n=5 ;;
    jun*) n=6 ;;
    jul*) n=7 ;;
    aug*) n=8 ;;
    sep*) n=9 ;;
```

```

    oct*) n=10 ;;
    nov*) n=11 ;;
    dec*) n=12 ;;
esac

echo ""

if [ $m -ge $n ]
then
    echo -e "Invalid range of month."
else
    set `date`
    y=`echo $4`
    for((i=m;i<=n;i++))
    do
        cal $i $y
    done
fi

echo "-----"

```

### Output:

```

ani@erna:~$ ./3calrange.sh
-----
Displaying calenders of the specified range of months.

Enter first month of range: Mar
Enter last month of range: May

    March 2020
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

    April 2020
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

    May 2020
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
-----

```

**Q4. Write a shell script to accept a login name. If not a valid login name display message - "Entered login name is invalid".**

**Code:**

```
echo "-----"
echo "Checking whether the entered login name is valid or not."
echo ""

echo -n "Enter a login name: " ; read log

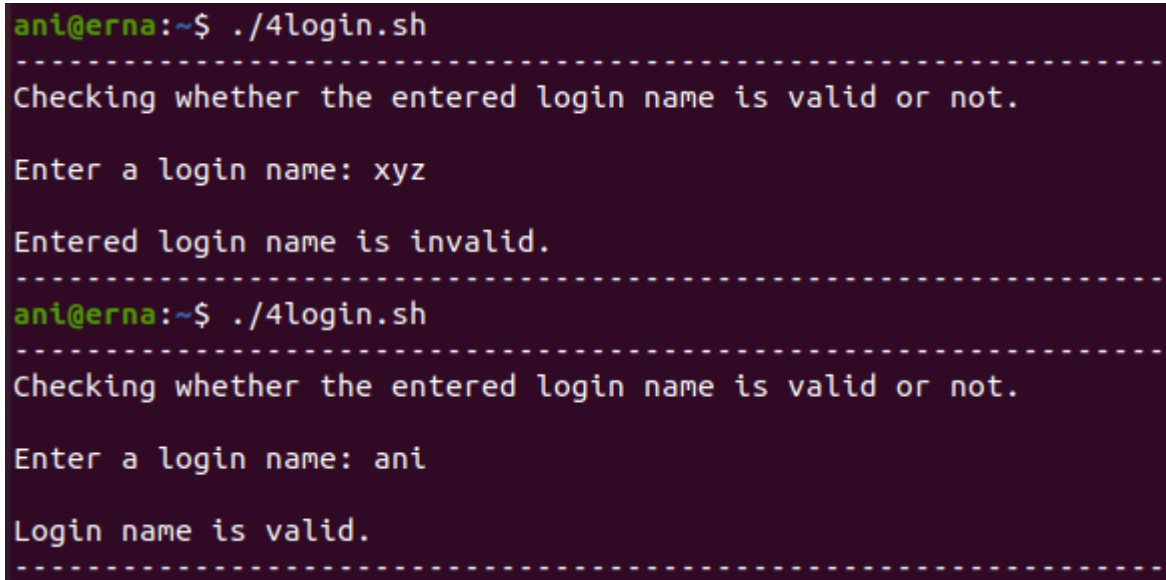
who -q -H | head -1 | grep "$log" > /dev/null

status=$?

echo ""

if [ $status -eq 1 ]
then
    echo "Entered login name is invalid."
else
    echo "Login name is Valid."
fi
echo "-----"
```

**Output:**



```
ani@erna:~$ ./4login.sh
-----
Checking whether the entered login name is valid or not.

Enter a login name: xyz

Entered login name is invalid.
-----
ani@erna:~$ ./4login.sh
-----
Checking whether the entered login name is valid or not.

Enter a login name: ani

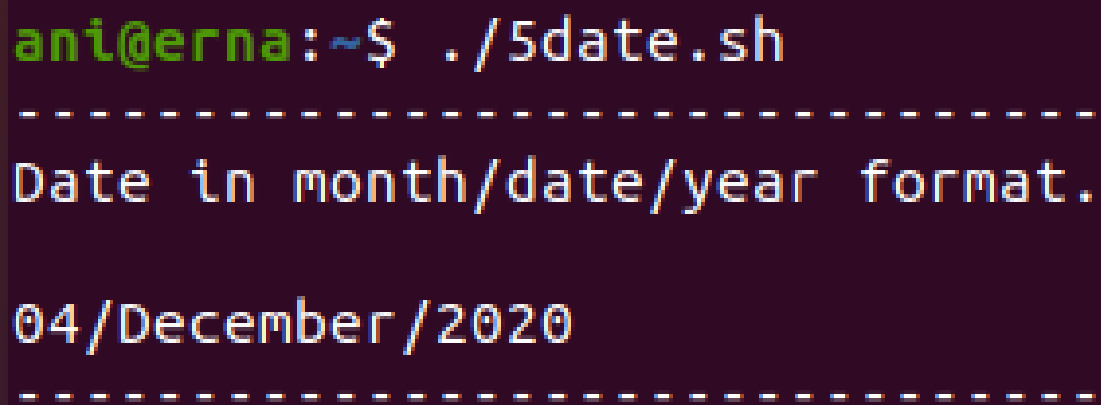
Login name is valid.
-----
```

**Q5. Write a shell script to display date in the mm/dd/yy format.**

**Code:**

```
echo "-----"  
echo "Date in month/date/year format."  
echo ""  
  
set `date`  
echo $2/$3/$6  
echo "-----"
```

**Output:**



The screenshot shows a terminal window with a dark background. The prompt is 'ani@erna:~\$'. The command './5date.sh' has been entered. The output consists of a dashed line, the text 'Date in month/date/year format.', the date '04/December/2020', and another dashed line.

```
ani@erna:~$ ./5date.sh  
-----  
Date in month/date/year format.  
04/December/2020  
-----
```



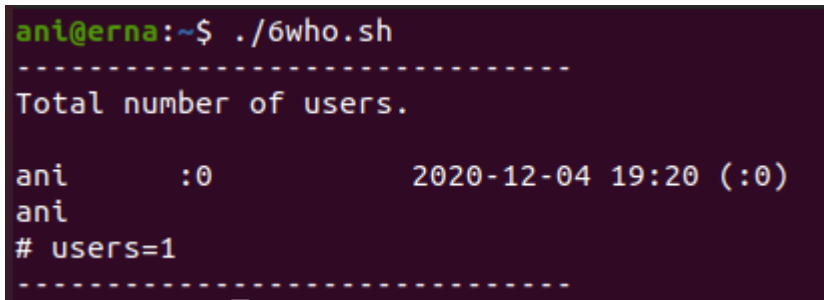
**Q6. Write a shell script to display on the screen sorted output of “who” command along with the total number of users.**

**Code:**

```
echo "-----"
echo "Total number of users."
echo ""

who | sort
who -q
echo "-----"
```

**Output:**



```
ani@erna:~$ ./6who.sh
-----
Total number of users.

ani      :0          2020-12-04 19:20 (:0)
ani
# users=1
-----
```

**Q7. Write a shell script to display the multiplication table any number.**

**Code:**

```
echo "-----"
echo "Program to print the multiplication table."
echo ""

echo -n "Enter a number for its multiplication table: "; read n
echo ""

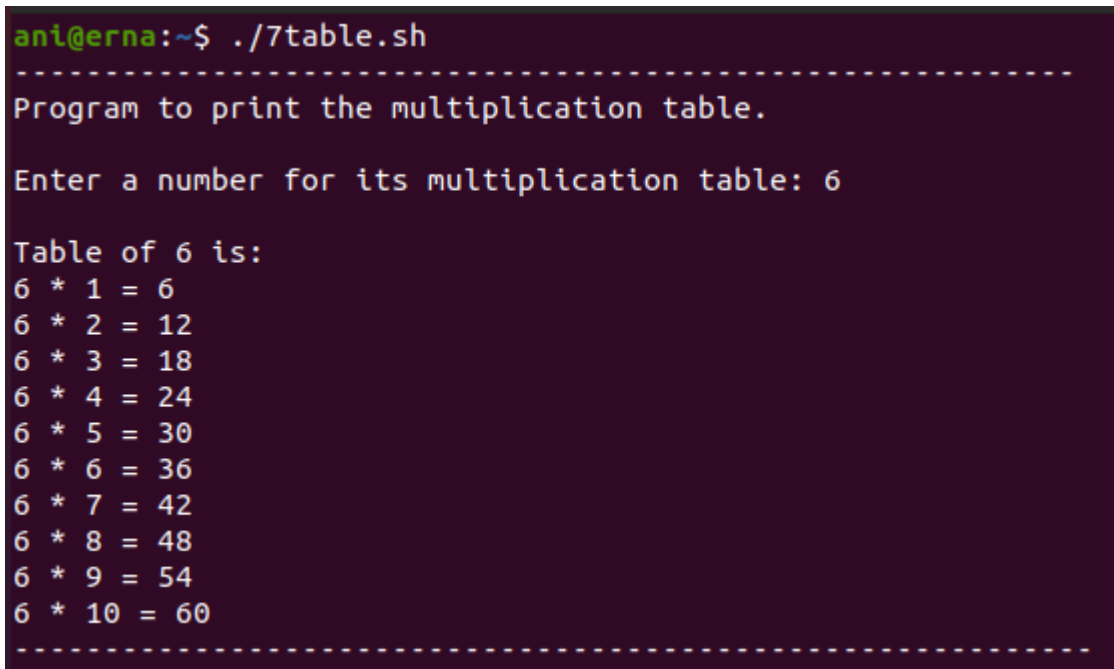
count=1

echo "Table of $n is: "

while [ $count -le 10 ]
do
    ((res=n * count))
    echo "$n * $count = $res"
    count=`expr $count + 1`
done

echo "-----"
```

**Output:**



The screenshot shows a terminal window with a dark background. The prompt is 'ani@erna:~\$'. The user has entered './7table.sh'. The script outputs a dashed line, followed by 'Program to print the multiplication table.', then another dashed line. It prompts 'Enter a number for its multiplication table: 6'. The output shows 'Table of 6 is:' followed by a list of multiplication results from 6 \* 1 to 6 \* 10. The list ends with a dashed line.

```
ani@erna:~$ ./7table.sh
-----
Program to print the multiplication table.
-----
Enter a number for its multiplication table: 6

Table of 6 is:
6 * 1 = 6
6 * 2 = 12
6 * 3 = 18
6 * 4 = 24
6 * 5 = 30
6 * 6 = 36
6 * 7 = 42
6 * 8 = 48
6 * 9 = 54
6 * 10 = 60
-----
```

**Q8. Write a shell script to compare two files and if found equal asks the user to delete the duplicate file.**

**Code:**

```
echo "-----"
echo "Program to compare 2 files & delete the duplicate one."
echo ""

echo -n "Enter name of file 1: " ; read f1
echo -n "Enter name of file 2: " ; read f2

if [ -s $f1 ]
then
    if [ -s $f2 ]
    then
        cmp $f1 $f2
        if [ $? -eq 0 ]
        then
            echo "Files are identical"
            echo -n "Do you want to delete the duplicate file? (y/n): "
            read ch
            case $ch in
                [yY]*) rm -rf $f2
                    echo "File deleted." ;;
                [nN]*) echo "Deletion Canceled."
                    exit 0 ;;
                *) echo "Invalid Choice!!"
                    exit 1
            esac
        else
            echo "Files are unique."
        fi
    else
        echo "$f2 is empty."
    fi
else
    echo "$f2 is empty."
fi
echo "-----"
```

### Output:

```
ani@erna:~$ echo hello>temp
ani@erna:~$ echo bye>temp1
ani@erna:~$ ./8compare.sh
-----
Program to compare 2 files & delete the duplicate one.

Enter name of file 1: temp
Enter name of file 2: temp1
temp temp1 differ: byte 1, line 1
Files are unique.
-----
ani@erna:~$ echo hello>temp
ani@erna:~$ echo hello>temp1
ani@erna:~$ ./8compare.sh
-----
Program to compare 2 files & delete the duplicate one.

Enter name of file 1: temp
Enter name of file 2: temp1
Files are identical
Do you want to delete the duplicate file? (y/n): y
File deleted.
-----
```

**Q9. Write a shell script to find the sum of digits of a given number.**

**Code:**

```
echo "-----"
echo "Finding sum of digit of the number."
echo ""

echo -n "Enter the number: " ; read n

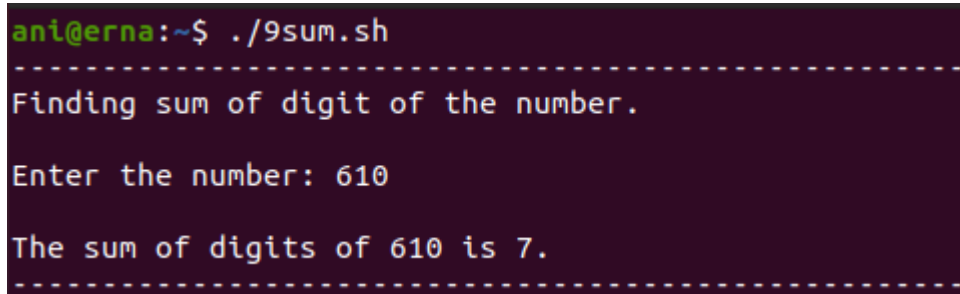
num=$n
sum=0

while [ $n -gt 0 ]
do
    ((rem=n%10))
    ((sum=sum+rem))
    ((n=n/10))
done

echo ""

echo "The sum of digits of $num is $sum."
echo "-----"
```

**Output:**



```
ani@erna:~$ ./9sum.sh
-----
Finding sum of digit of the number.

Enter the number: 610

The sum of digits of 610 is 7.
-----
```

**Q10. Write a shell script to merge the contents of three files, sort the contents and then display them page by page.**

**Code:**

```
echo "-----"
echo "Program to merge content of 3 files & then sort the content & display them page by
page."
echo ""

echo -n "Enter name of file 1: " ; read f1
echo -n "Enter name of file 2: " ; read f2
echo -n "Enter name of file 3: " ; read f3

if [ -f $f1 -a -f $f2 -a -f $f3 ]
then
    sort $f1 > t1
    sort $f2 > t2
    sort $f3 > t3
    echo -e "The sorted contents are as follows"
    sort -m t1 t2 t3 > sortemp.txt
    rm t1 t2 t3
    cat sortemp.txt | more | less
else
    echo "Files with these names does not exist."
    echo "-----"
fi
```

## Output:

```
ani@erna:~$ ./10merge.sh
-----
Program to merge content of 3 files & then sort the content & display them page by page.

Enter name of file 1: 4login.sh
Enter name of file 2: 5date.sh
Enter name of file 3: 7table.sh
The sorted contents are as follows

count=1
  count=`expr $count + 1`
do
done
echo ""
echo ""
echo ""
echo ""
echo ""
echo "-----"
echo "-----"
echo "-----"
echo "-----"
echo "-----"
echo "$2/$3/$4
  echo "$n * $count = $res"
echo "Checking whether the entered login name is valid or not."
echo "Date in month/date/year format."
  echo "Entered login name is invalid."
  echo "Login name is valid."
echo -n "Enter a login name: " ; read log
echo -n "Enter a number for its multiplication table: " ; read n
echo "Program to print the multiplication table."
echo "Table of $n is: "
else
fi
if [ $status -eq 1 ]
  ((res=n * count))
set `date`
status=$?
then
while [ $count -le 10 ]
who -q -H | head -1 | grep "$log" > /dev/null
(END)
```

**Q11. Write a shell script to find the LCD (least common divisor) of two numbers.**

**Code:**

```
echo "-----"
echo "Program to find out the LCD of given numbers."
echo ""
echo -n "Enter Number 1 : ";read a
echo -n "Enter Number 2 : ";read b

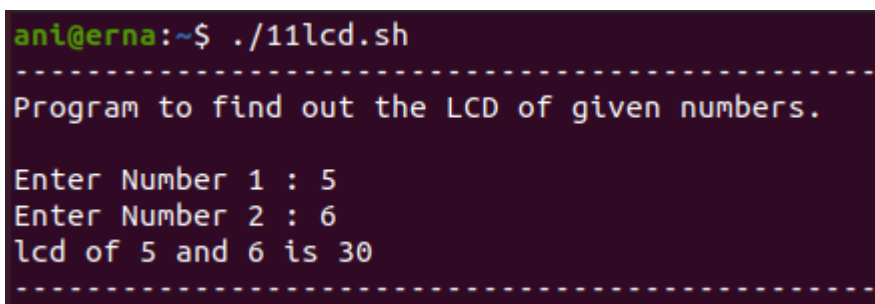
m=$a

if [ $b -gt $m ]
then
  m=$b
fi

while [ $m -ne 0 ]
do
  ((x = m % a))
  ((y = m % b))

  if [ $x -eq 0 -a $y -eq 0 ]
  then
    echo lcd of $a and $b is $m
    break
  fi
  m=`expr $m + 1`
done
echo "-----"
```

**Output:**



```
ani@erna:~$ ./11lcd.sh
-----
Program to find out the LCD of given numbers.

Enter Number 1 : 5
Enter Number 2 : 6
lcd of 5 and 6 is 30
-----
```



**Q12. Write a shell script to perform the tasks of basic calculator.**

**Code:**

```
echo "-----"
echo "Basic calculator."
echo ""

function operation()
{
case $ch in
1) res=`echo $a + $b | bc`
  echo "$a+$b"
  echo ;;
2) res=`echo $a - $b | bc`
  echo "$a-$b"
  echo ;;
3) res=`echo $a \* $b | bc`
  echo "$a*$b"
  echo ;;
4) if [ $b -eq 0 ]
  then
    echo "Denominator can't be 0."
    echo "Enter the numbers again!!"
    echo ""
    main
  else
    res=`echo "scale=2; $a/$b" | bc`
    echo "$a/$b"
  fi ;;
*) res="f"
esac
if [ $res == "t" ]
then
  echo "Program stopped on user request."
  exit
elif [ $res == "f" ]
then
  echo "Invalid Choice. Choose again."
  choice
else
  echo "Result: $res"
  echo
fi
contornot
}
function contornot()
{
echo -n "Do you want to continue(Y/N) : " ; read n
case $n in
```

```

[yY]* ) main ;;
[nN]* ) exit ;;
* ) echo "Wrong choice"
contornot
esac
}
function choice()
{
echo ""
echo "---MENU---"
echo "1. Addition"
echo "2. Subtraction"
echo "3. Multiplication"
echo "4. Division"
echo ""
echo -n "Enter your choice: " ; read ch
echo ""
operation
}

function main()
{
echo -n "Enter number 1: " ; read a
echo -n "Enter number 2: " ; read b
choice
}
main
echo "-----"

```

### Output:

```
ani@erna:~$ ./12calculator.sh
-----
Basic calculator.

Enter number 1: 610
Enter number 2: 2002

---MENU---
1. Addition
2. Subtraction
3. Multiplication
4. Division

Enter your choice: 3

610*2002

Result: 1221220

Do you want to continue(Y/N) : Y
Enter number 1: 1646153
Enter number 2: 48

---MENU---
1. Addition
2. Subtraction
3. Multiplication
4. Division

Enter your choice: 4

1646153/48
Result: 34294.85

Do you want to continue(Y/N) : N
ani@erna:~$
```

**Q13. Write a shell script to find the power of a given number.**

**Code:**

```
echo "-----"
echo "Program to find the solution of n raise to power n."
echo ""

echo -n "Enter the number: " ; read n
echo -n "Enter its power: " ; read p

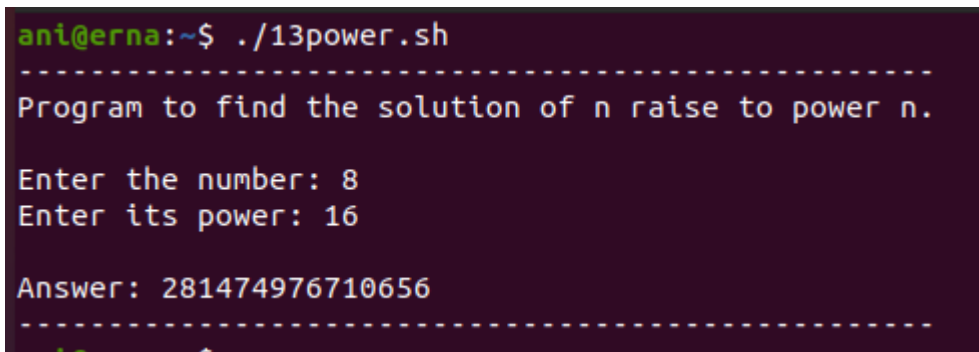
ans=1

for ((m=1 ; m<=p ; m++))
do
    ((ans=ans*n))
done

echo ""
echo "Answer: $ans"

echo "-----"
```

**Output:**



```
ani@erna:~$ ./13power.sh
-----
Program to find the solution of n raise to power n.

Enter the number: 8
Enter its power: 16

Answer: 281474976710656
-----
ani@erna:~$
```

#### 14. Write a shell script to find the factorial of a given number.

##### Code:

```
echo "-----"
echo "Program to find the factorial of a number."
echo ""

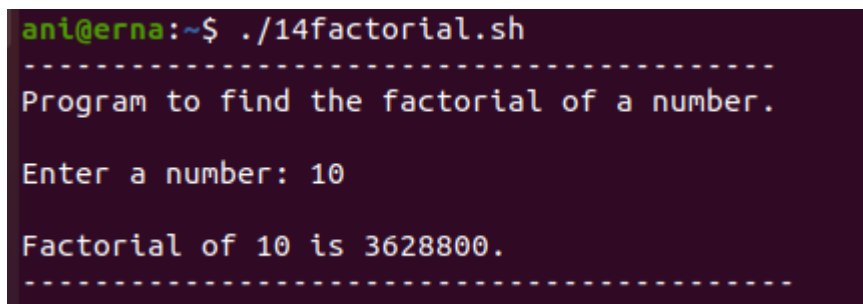
echo -n "Enter a number: " ; read n

num=$n
fact=1

while [ $n -ge 1 ]
do
    ((fact=fact*n))
    ((n=n-1))
done

echo ""
echo "Factorial of $num is $fact."
echo "-----"
```

##### Output:

A terminal window with a dark purple background. The prompt is 'ani@erna:~\$'. The user has entered './14factorial.sh'. The script outputs a dashed line, the title 'Program to find the factorial of a number.', a blank line, and the prompt 'Enter a number:'. The user has entered '10'. The script outputs 'Factorial of 10 is 3628800.' followed by another dashed line.

```
ani@erna:~$ ./14factorial.sh
-----
Program to find the factorial of a number.

Enter a number: 10

Factorial of 10 is 3628800.
-----
```

**Q15. Write a shell script to check whether the number is Armstrong or not.**

**Code:**

```
echo "-----"
echo "Finding whether the number is armstrong or not."
echo ""

echo -n "Enter a number: " ; read n

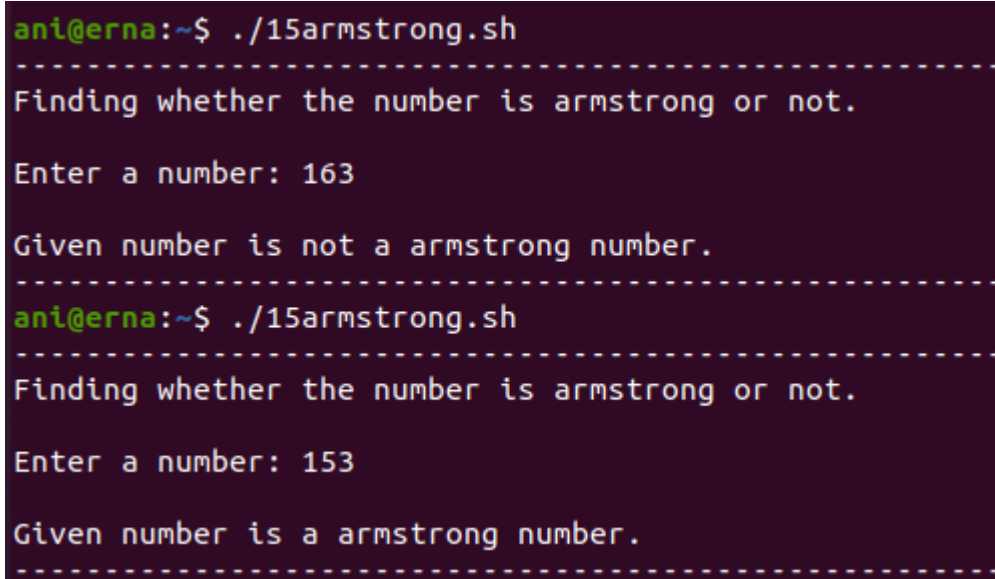
num=$n
sum=0

while [ $n -gt 0 ]
do
    ((rem=n%10))
    ((sum=sum + rem*rem*rem))
    ((n=n/10))
done

echo ""

if [ $sum -eq $num ]
then
    echo "Given number is a armstrong number."
else
    echo "Given number is not a armstrong number."
fi
echo "-----"
```

**Output:**



```
ani@erna:~$ ./15armstrong.sh
-----
Finding whether the number is armstrong or not.

Enter a number: 163

Given number is not a armstrong number.
-----
ani@erna:~$ ./15armstrong.sh
-----
Finding whether the number is armstrong or not.

Enter a number: 153

Given number is a armstrong number.
-----
```

**Q16. Write a shell script to check whether the file have all the permissions or not.**

**Code:**

```
echo "-----"
echo "Checking whether the file have all the permissions or not."
echo ""

echo -n "Enter the file name: " ; read f
echo ""

if [ -e $f ]
then
    echo "File Exists"
else
    echo "File does not exists"
    exit
fi

echo ""
echo "File Permissions: "

if [ -w $f ]
then
    echo "Writable"
else
    echo "Not Writable"
fi

if [ -r $f ]
then
    echo "Readable"
else
    echo "Not Readable"
fi

if [ -x $f ]
then
    echo "Executable"
else
    echo "Not Executable"
fi
echo "-----"
```

### Output:

```
ani@erna:~$ chmod 333 temp1
ani@erna:~$ chmod 555 temp2
ani@erna:~$ chmod 777 temp3
ani@erna:~$ ./16permission.sh
-----
Checking whether the file have all the permissions or not.

Enter the file name: temp1

File Exists

File Permissions:
Writable
Not Readable
Executable
-----
ani@erna:~$ ./16permission.sh
-----
Checking whether the file have all the permissions or not.

Enter the file name: temp2

File Exists

File Permissions:
Not Writable
Readable
Executable
-----
ani@erna:~$ ./16permission.sh
-----
Checking whether the file have all the permissions or not.

Enter the file name: temp3

File Exists

File Permissions:
Writable
Readable
Executable
-----
```



**Q17. Write a shell script to show the “Pyramid” of special character “\*”.**

**Code:**

```
echo "-----"
echo "Program to print pyramid."
echo ""

echo -n "Enter the no. of rows in a pyramid: " ; read n

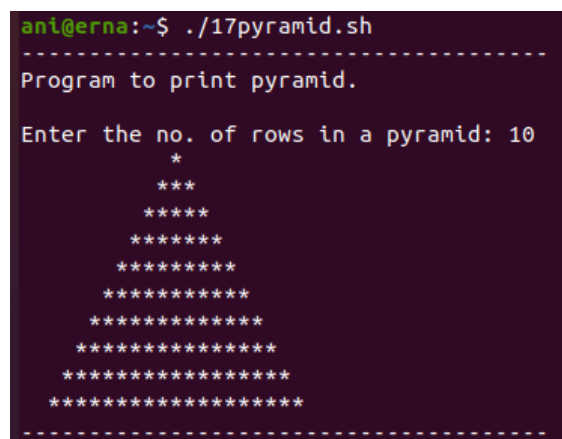
for((m=1;m<=n;m++))
do
    for((a=i;a<=n;a++))
    do
        echo -ne " "
    done

    for((p=1;p<=m;p++))
    do
        echo -ne "*"
    done

    for((i=1;i<m;i++))
    do
        echo -ne "*"
    done

    echo ""
done
echo "-----"
```

**Output:**



```
ani@erna:~$ ./17pyramid.sh
-----
Program to print pyramid.

Enter the no. of rows in a pyramid: 10
  *
 ***
*****
*****
*****
*****
*****
*****
*****
*****
*****
-----
```