UNIVERSITY OF CALCUTTA

SHELL POGRAMMING

Name : Anit Halder

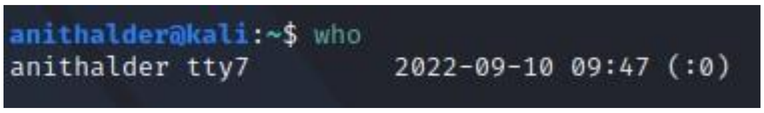
Roll : 213513-21-0115

Reg . No.: 513-1112-21-0459

SHELL COMMAND

1. ‘ who ‘ command

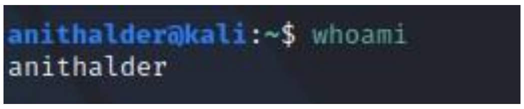
The main usage of **who** command in Linux without command-line parameter is to show the name of the users who are logged in currently.



```
anithalder@kali:~$ who
anithalder tty7         2022-09-10 09:47 (:0)
```

2. ‘ whoami ‘ command

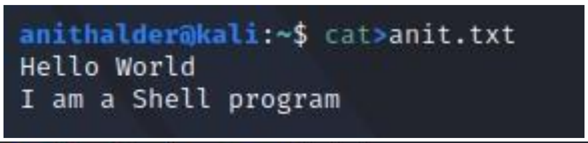
The **whoami** command allows Linux users to see the currently logged-in user. The output displays the username of the effective user in the current shell.Additionally, **whoami** is useful in bash scripting to show who is running the script .



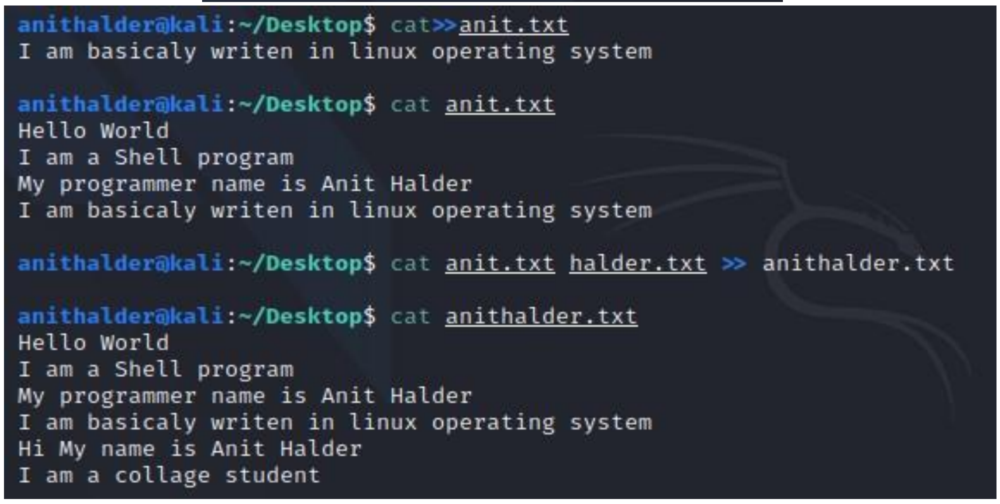
```
anithalder@kali:~$ whoami
anithalder
```

3. ‘ cat ‘ command

The **cat** (concatenate) command is one of the most frequently used commands in Linux/Unix-like operating systems. **cat** command allows us to create single or multiple files, view content of a file, concatenate files and redirect output in terminal or files.



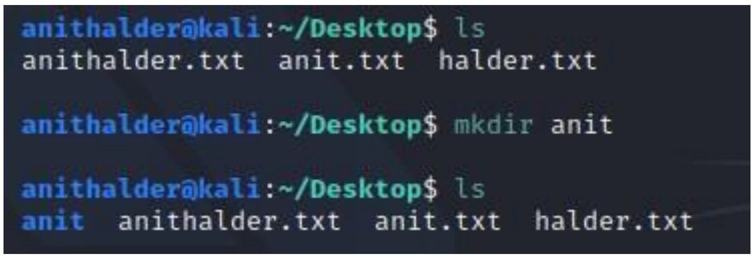
```
anithalder@kali:~$ cat>anit.txt
Hello World
I am a Shell program
```



```
anithalder@kali:~/Desktop$ cat>>anit.txt
I am basicaly writen in linux operating system

anithalder@kali:~/Desktop$ cat anit.txt
Hello World
I am a Shell program
My programmer name is Anit Halder
I am basicaly writen in linux operating system

anithalder@kali:~/Desktop$ cat anit.txt halder.txt >> anithalder.txt

anithalder@kali:~/Desktop$ cat anithalder.txt
Hello World
I am a Shell program
My programmer name is Anit Halder
I am basicaly writen in linux operating system
Hi My name is Anit Halder
I am a collage student
```

4. ‘ mkdir ‘ command

The mkdir command in Linux allows users to create or make new directories. mkdir stands for “make directory.” With mkdir , you can also set permissions, create multiple directories (folders) at once, and much more.

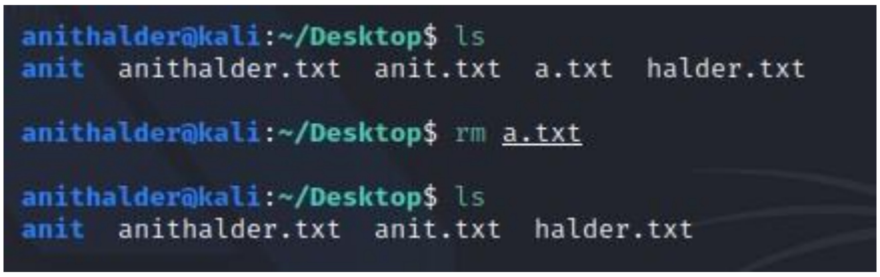


```
anithalder@kali:~/Desktop$ ls
anithalder.txt  anit.txt  halder.txt

anithalder@kali:~/Desktop$ mkdir anit

anithalder@kali:~/Desktop$ ls
anit  anithalder.txt  anit.txt  halder.txt
```

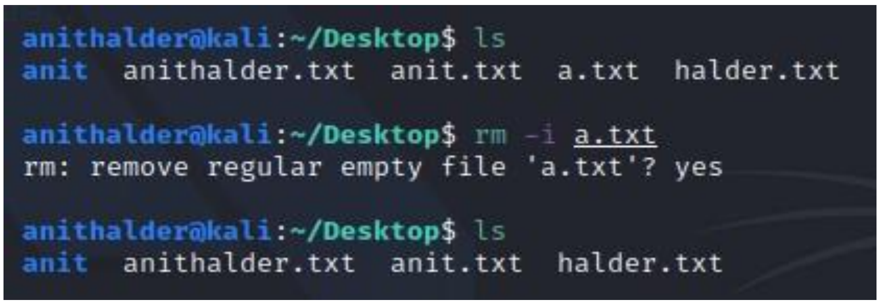
5. ‘ rm ‘ command

The rm command removes the entries for a specified file, group of files, or certain select files from a list within a directory. User confirmation, read permission, and write permission are not required before a file is removed when you use the rm command.

-f **:** Forces the removal of all files or directories.
-i **:** Prompts for confirmation before removing.
-I **:** Prompts once before removing more than three files or when removing recursively.
-r **:** Removes directories and their content recursively.
-d **:** Removes empty directories.
-v **:** Provides a verbose output.



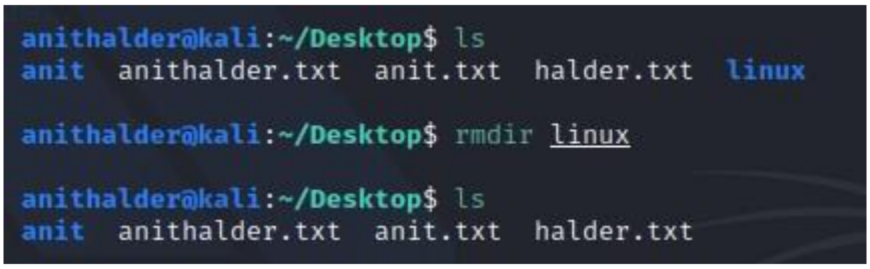
```
anithalder@kali:~/Desktop$ ls
anit  anithalder.txt  anit.txt  a.txt  halder.txt

anithalder@kali:~/Desktop$ rm a.txt

anithalder@kali:~/Desktop$ ls
anit  anithalder.txt  anit.txt  halder.txt
```



```
anithalder@kali:~/Desktop$ ls
anit  anithalder.txt  anit.txt  a.txt  halder.txt

anithalder@kali:~/Desktop$ rm -i a.txt
rm: remove regular empty file 'a.txt'? yes

anithalder@kali:~/Desktop$ ls
anit  anithalder.txt  anit.txt  halder.txt
```

6. ‘ rmdir ‘ command

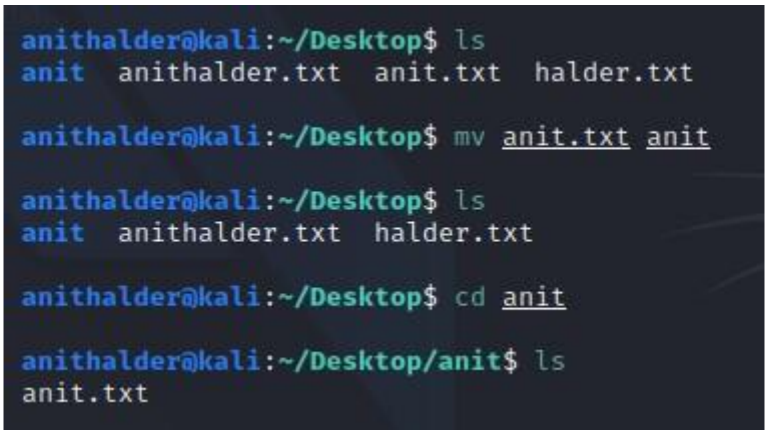
`rmdir` command removes empty directories specified on the command line. These directories are removed from the filesystem in the same order as they are specified on the command line, i.e., from left to right. If the directory is not empty you get an error message.



```
anithalder@kali:~/Desktop$ ls
anit  anithalder.txt  anit.txt  halder.txt  linux

anithalder@kali:~/Desktop$ rmdir linux

anithalder@kali:~/Desktop$ ls
anit  anithalder.txt  anit.txt  halder.txt
```

7. ‘ mv ‘ command

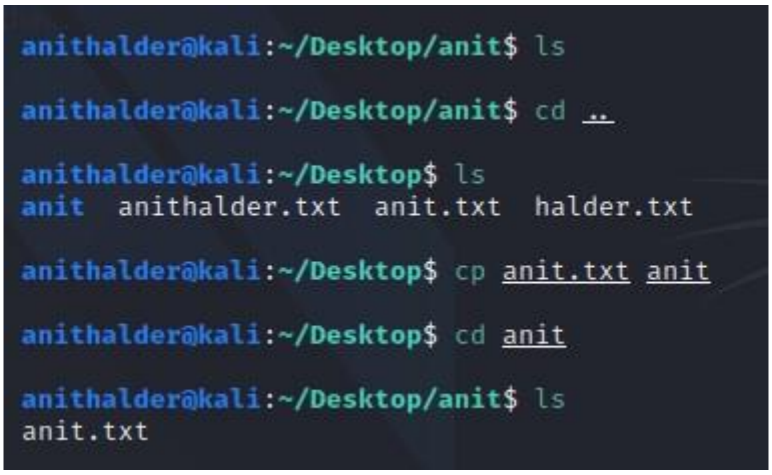
The `mv` command to move files and directories from one directory to another or to rename a file or directory. If you move a file or directory to a new directory without specifying a new name, it retains its original name.



```
anithalder@kali:~/Desktop$ ls
anit  anithalder.txt  anit.txt  halder.txt

anithalder@kali:~/Desktop$ mv anit.txt anit

anithalder@kali:~/Desktop$ ls
anit  anithalder.txt  halder.txt

anithalder@kali:~/Desktop$ cd anit

anithalder@kali:~/Desktop/anit$ ls
anit.txt
```

8. ‘ cp ‘ command

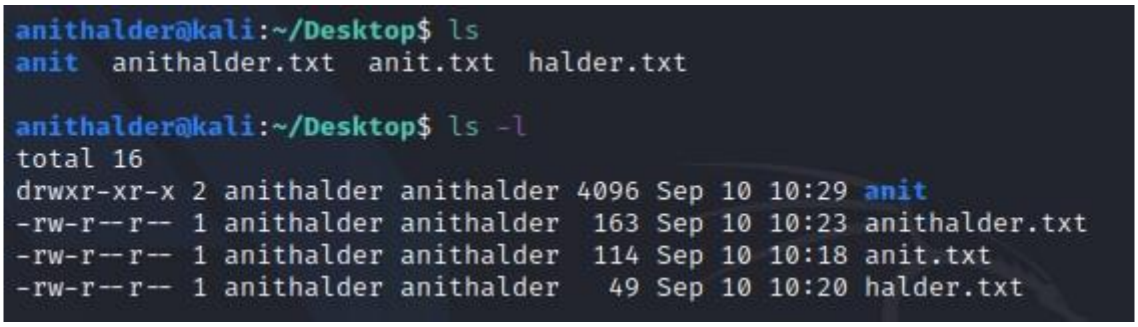
We use the `cp` command for copying files from one location to another. This command can also copy directories (folders).



```
anithalder@kali:~/Desktop/anit$ ls

anithalder@kali:~/Desktop/anit$ cd ..

anithalder@kali:~/Desktop$ ls
anit  anithalder.txt  anit.txt  halder.txt

anithalder@kali:~/Desktop$ cp anit.txt anit

anithalder@kali:~/Desktop$ cd anit

anithalder@kali:~/Desktop/anit$ ls
anit.txt
```

9. ‘ ls ‘ command

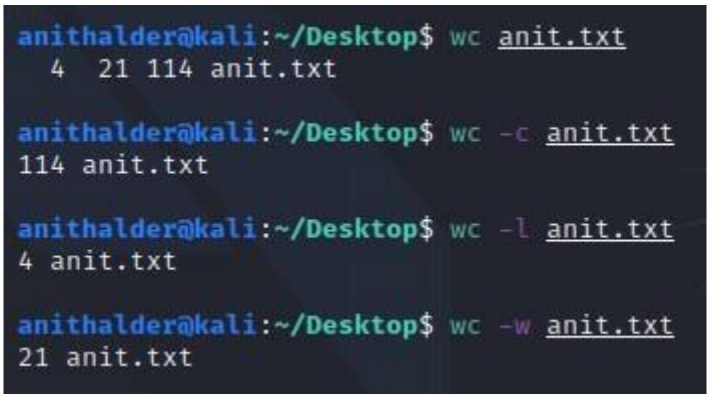
The **ls** command is used to list files or directories in Linux and other Unix-based operating systems. Just like you navigate in your File explorer or Finder with a GUI, the **ls** command allows you to list all files or directories in the current directory by default, and further interact with them via the command line.



```
anithalder@kali:~/Desktop$ ls
anit  anithalder.txt  anit.txt  halder.txt

anithalder@kali:~/Desktop$ ls -l
total 16
drwxr-xr-x 2 anithalder anithalder 4096 Sep 10 10:29 anit
-rw-r--r-- 1 anithalder anithalder  163 Sep 10 10:23 anithalder.txt
-rw-r--r-- 1 anithalder anithalder  114 Sep 10 10:18 anit.txt
-rw-r--r-- 1 anithalder anithalder   49 Sep 10 10:20 halder.txt
```

10. ‘ wc ‘ command

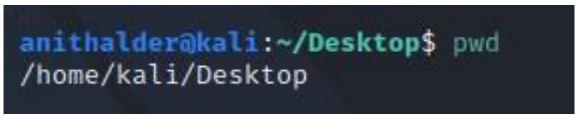
The **wc** command in Linux is a command line utility for printing newline, word and byte counts for files. It can return the number of lines in a file, the number of characters in a file and the number of words in a file. It can also be combine with pipes for general counting operations.



```
anithalder@kali:~/Desktop$ wc anit.txt
  4  21 114 anit.txt

anithalder@kali:~/Desktop$ wc -c anit.txt
114 anit.txt

anithalder@kali:~/Desktop$ wc -l anit.txt
4 anit.txt

anithalder@kali:~/Desktop$ wc -w anit.txt
21 anit.txt
```

11. ‘ pwd ‘ command

The **pwd** command writes to standard output the full path name of your current directory (from the root directory). All directories are separated by a / (slash). The root directory is represented by the first /, and the last directory named is your current directory.

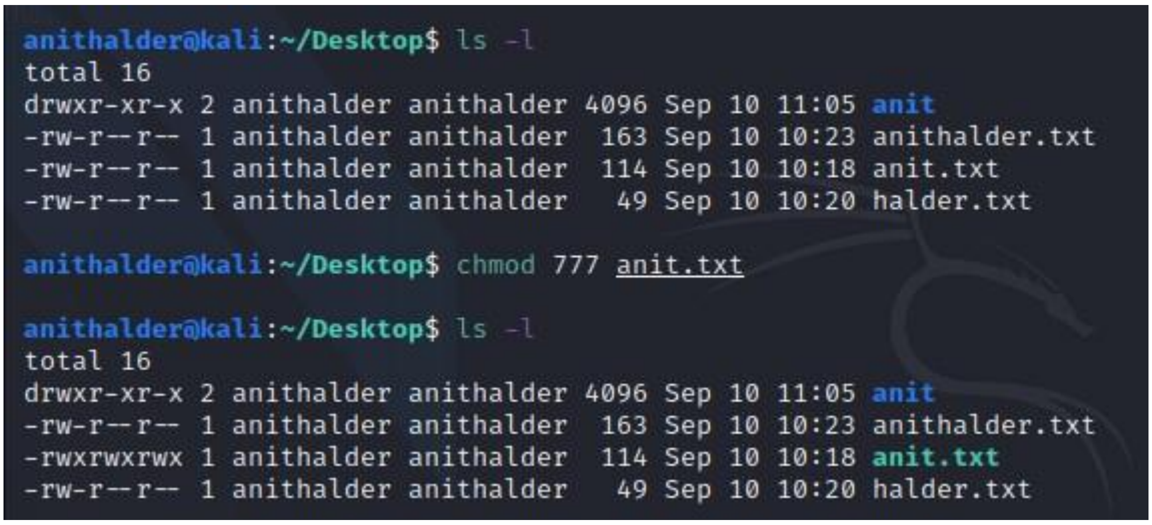


```
anithalder@kali:~/Desktop$ pwd
/home/kali/Desktop
```

12. ‘ chmod ‘ command

The **chmod** (short for change mode) command is used to manage file system access permissions on Linux and Unix-like systems. There are three basic file system permissions, or modes, to files and directories: read (r) write (w).

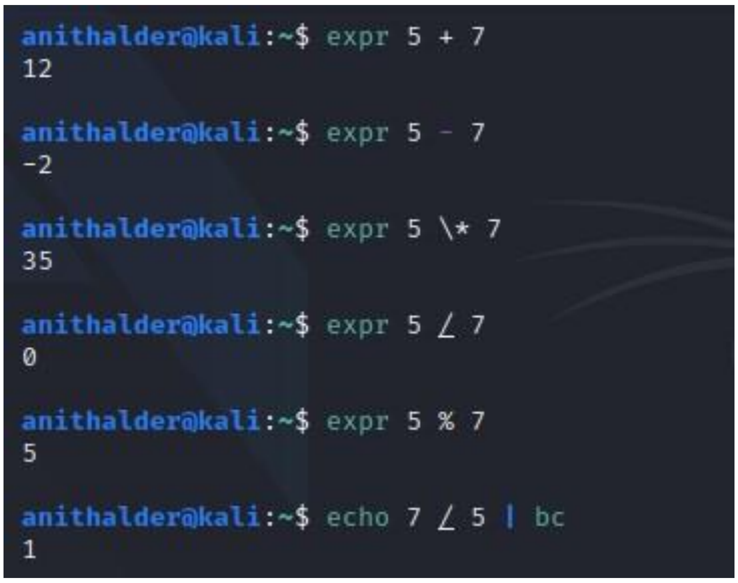
- chmod +rwx filename to add permissions.
- chmod -rwx directoryname to remove permissions.
- chmod +x filename to allow executable permissions.
- chmod -wx filename to take out write and executable permissions.



```
anithalder@kali:~/Desktop$ ls -l
total 16
drwxr-xr-x 2 anithalder anithalder 4096 Sep 10 11:05 anit
-rw-r--r-- 1 anithalder anithalder  163 Sep 10 10:23 anithalder.txt
-rw-r--r-- 1 anithalder anithalder  114 Sep 10 10:18 anit.txt
-rw-r--r-- 1 anithalder anithalder   49 Sep 10 10:20 halder.txt

anithalder@kali:~/Desktop$ chmod 777 anit.txt

anithalder@kali:~/Desktop$ ls -l
total 16
drwxr-xr-x 2 anithalder anithalder 4096 Sep 10 11:05 anit
-rw-r--r-- 1 anithalder anithalder  163 Sep 10 10:23 anithalder.txt
-rwxrwxrwx 1 anithalder anithalder  114 Sep 10 10:18 anit.txt
-rw-r--r-- 1 anithalder anithalder   49 Sep 10 10:20 halder.txt
```

13. ‘ expr ‘ command

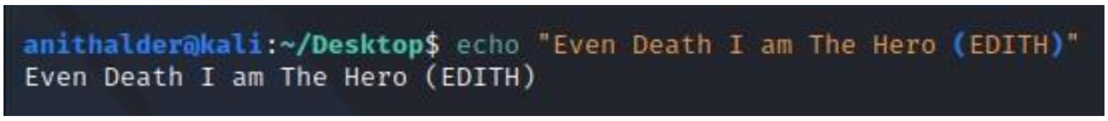
expr is a command line Linux utility which evaluates an expression and outputs the corresponding value. **expr** evaluates integer or string expressions, including pattern matching regular expressions.



```
anithalder@kali:~$ expr 5 + 7
12

anithalder@kali:~$ expr 5 - 7
-2

anithalder@kali:~$ expr 5 \* 7
35

anithalder@kali:~$ expr 5 / 7
0

anithalder@kali:~$ expr 5 % 7
5

anithalder@kali:~$ echo 7 / 5 | bc
1
```

14. ‘ echo ‘ command

echo command in linux is used to display line of text/string that are passed as an argument . This is a built in command that is mostly used in shell scripts and batch files to output status text to the screen or a file. In above example, text after \c is not printed and omitted trailing new line.

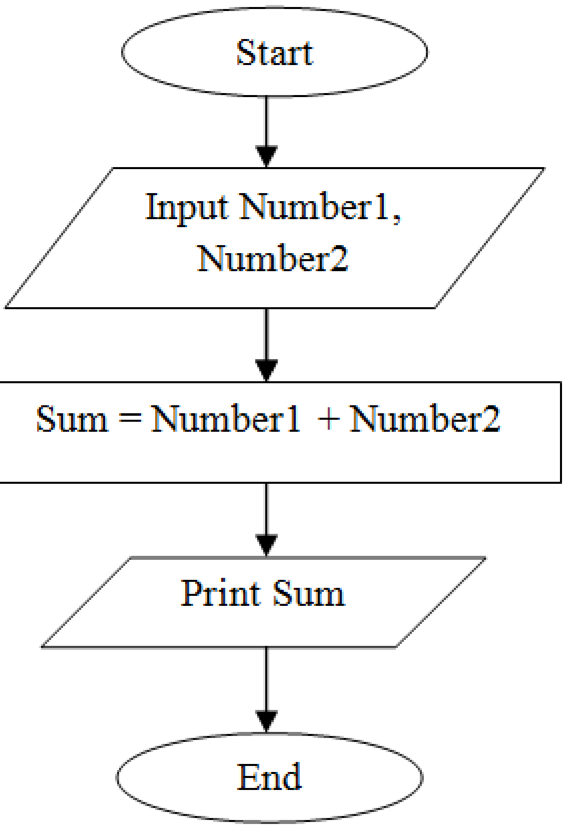


```
anithalder@kali:~/Desktop$ echo "Even Death I am The Hero (EDITH)"
Even Death I am The Hero (EDITH)
```

SHELL SCRIPTING

1. Write a shell program to perform the addition of two numbers.

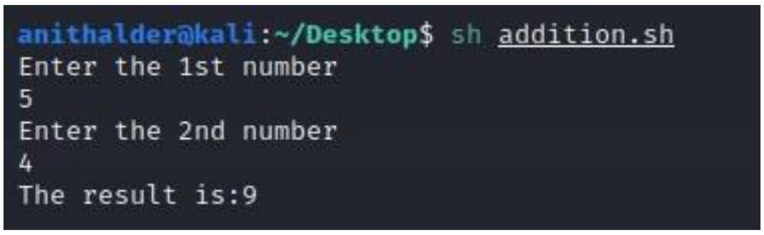
Flowchart :



Source Code :

```
echo "Enter the 1st number"
read a
echo "Enter the 2nd number"
read b
S=`expr $a + $b`
echo "The result is:"$S
```

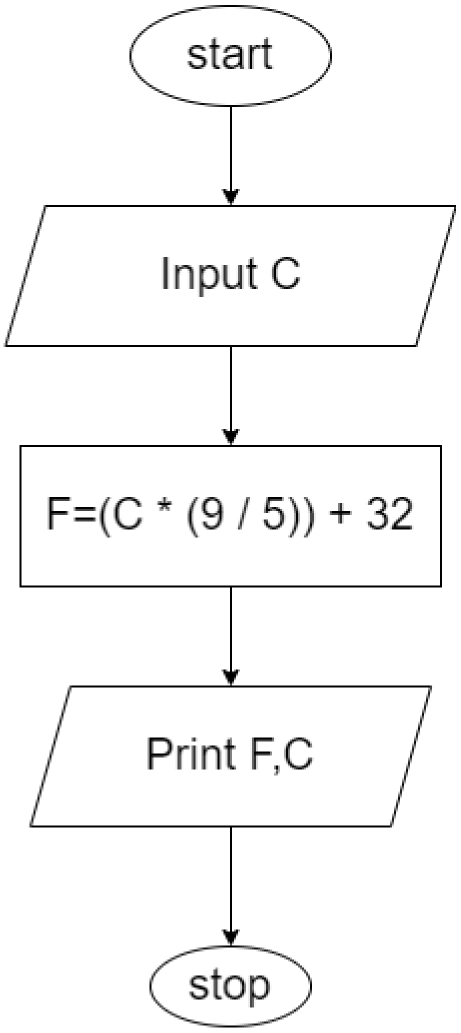
Output :



```
anithalder@kali:~/Desktop$ sh addition.sh
Enter the 1st number
5
Enter the 2nd number
4
The result is:9
```

2. Write a shell program to convert Centrigrate to Farenhight.

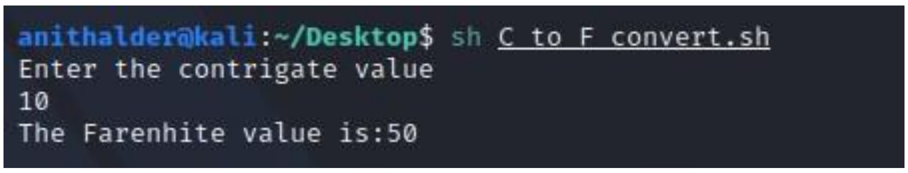
- **Flowchart** :



- **Source Code** :

```
echo "Enter the contrigate value"
read c
F=`expr $c \* 9 / 5 + 32`
echo "The Farenhite value is:"$F
```

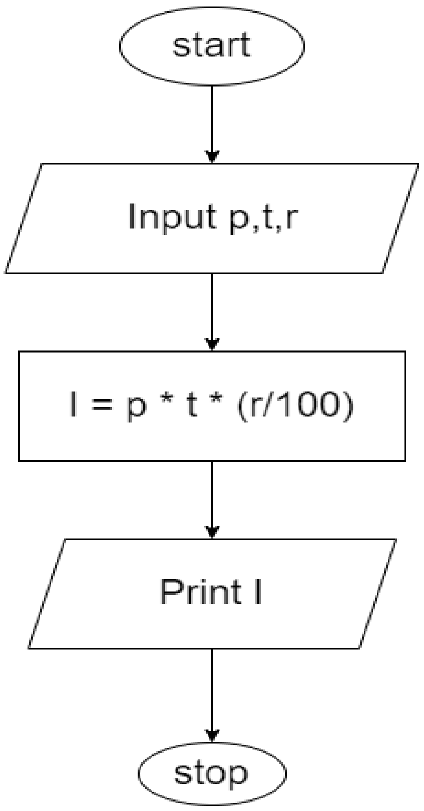
- **Output** :



```
anithalder@kali:~/Desktop$ sh C to F convert.sh
Enter the contrigate value
10
The Farenhite value is:50
```

3. Write a shell script to calculate simple interest.

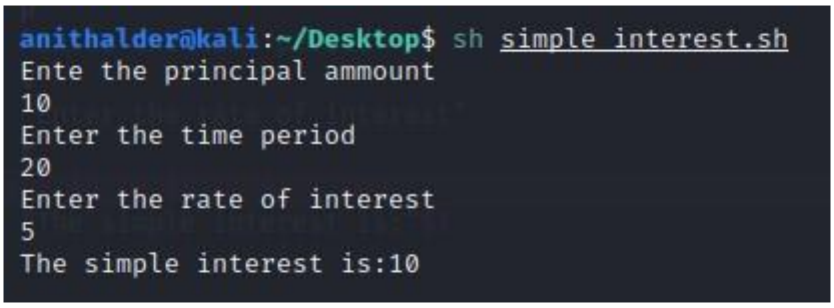
Flowchart :



Source Code :

```
echo "Ente the principal ammount"
read p
echo "Enter the time period"
read t
echo "Enter the rate of interest"
read r
I=`expr $p \* $t \* $r / 100`
echo "The simple interest is:"$I
```

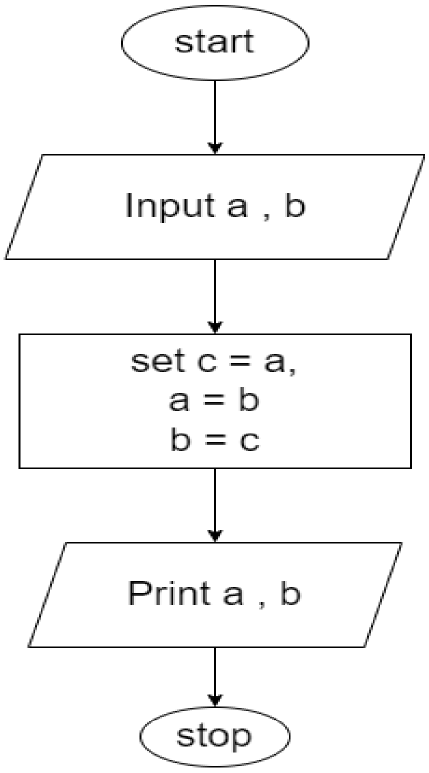
Output :



```
anithalder@kali:~/Desktop$ sh simple_interest.sh
Ente the principal ammount
10
Enter the time period
20
Enter the rate of interest
5
The simple interest is:10
```

4. Write a shell script to swaping of two numbers.

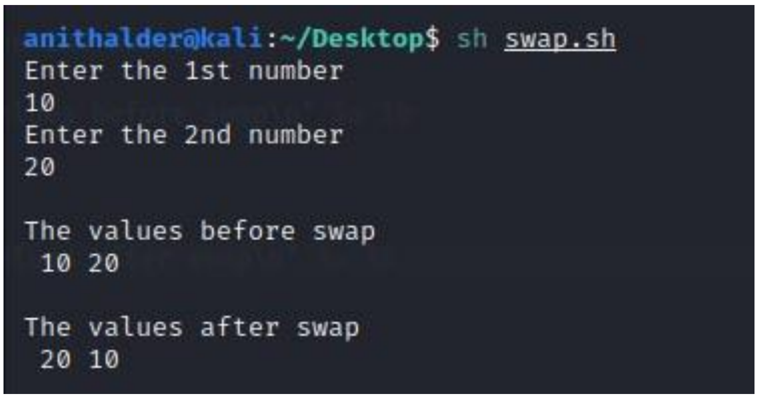
Flowchart :



Source Code :

```
echo "Enter the 1st number"
read a
echo "Enter the 2nd number"
read b
echo "\nThe values before swap\n" $a $b
c=$a
a=$b
b=$c
echo "\nThe values after swap\n" $a $b
```

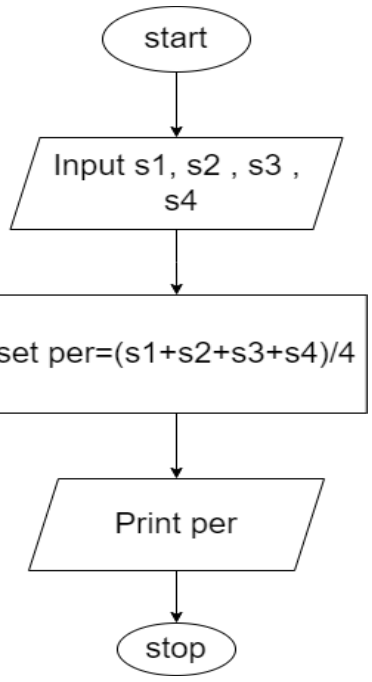
Output :



```
anithalder@kali:~/Desktop$ sh swap.sh
Enter the 1st number
10
Enter the 2nd number
20

The values before swap
 10 20

The values after swap
 20 10
```

5. Write a shell script to calculate percentage(%) of previous semester mark.

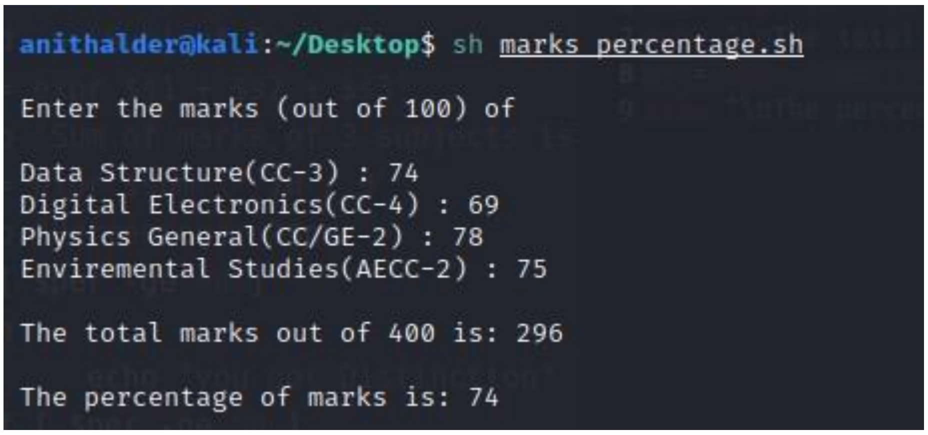
Flowchart :



Source Code :

```
echo "\nEnter the marks (out of 100) of\n"
read -p "Data Structure(CC-3) : " a
read -p "Digital Electronics(CC-4) : " b
read -p "Physics General(CC/GE-2) : " c
read -p "Envirmental Studies(AECC-2) : " d
sum=`expr $a + $b + $c + $d`
echo "\nThe total marks out of 400 is: "$sum
per=`expr $sum / 4`
echo "\nThe percentage of marks is: "$per
```

Output :



```
anithalder@kali:~/Desktop$ sh marks_percentage.sh

Enter the marks (out of 100) of

Data Structure(CC-3) : 74
Digital Electronics(CC-4) : 69
Physics General(CC/GE-2) : 78
Envirementa Studies(AECC-2) : 75

The total marks out of 400 is: 296

The percentage of marks is: 74
```

6. Write a shell program to find the maximum number between two number.

Algorithm Maximum_Two

Input : Read two vlues from the user
Output : Print the greater number between two numbrs

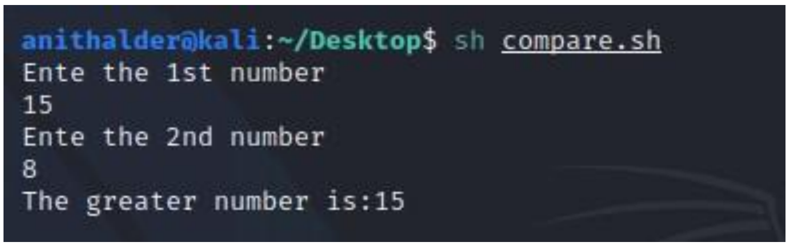
Steps :

```
1. Start
2. If (a>b) then
   2.1.  Print "The greater value is" a
3. Else
   3.1.  Print "The greater value is" b
4. End if
5. Stop
```

Source code :

```
echo "Ente the 1st number"
read a
echo "Ente the 2nd number"
read b
if [ $a -gt $b ]
then
echo "The greater number is:"$a
else
echo "The greater number is:"$b
fi
```

Output :



```
anithalder@kali:~/Desktop$ sh compare.sh
Ente the 1st number
15
Ente the 2nd number
8
The greater number is:15
```

7. Write a shell script program to calculate the what is the greater number between three number.

Algorithm

Input : Enter the three numbers which given by the user.
Output : Print the greater number in between the three numbers.

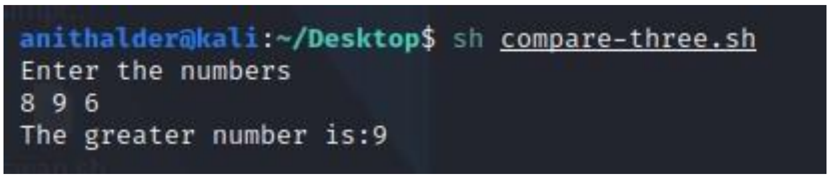
Steps :

```
1. Start
2. Read three numbers a,b & c    // a,b & c are user given value
3. If (a>b)
       3.1. if (a>c) then
               3.1.1. Print a
       3.2. else
               3.2.1. Print c
4. Else
       4.1.  if (b>c) then
               4.1.1. Print b
       4.2. else
               4.2.1. Print c
5. End if
6. Stop
```

Sorce code :

```
echo "Enter the numbers"
read a b c
if [ $a -gt $b ]
then
        if [ $a -gt $c ]
        then
                echo "The greater number is:"$a
                else
                echo "The greater number is:"$c
        fi
elif [ $a -eq $b -a $b -eq $c ]
then
echo "The number are equal"
else
        if [ $b -gt $c ]
        then
                echo "The greater number is:"$b
                else
                echo "The greater number is:"$c
        fi
fi
```

Output :



```
anithalder@kali:~/Desktop$ sh compare-three.sh
Enter the numbers
8 9 6
The greater number is:9
```

8. Write a shell script program to calculate the number is even or odd.

Algorithm

Input : Read a number from the user.
Output : Print the number is even or odd.

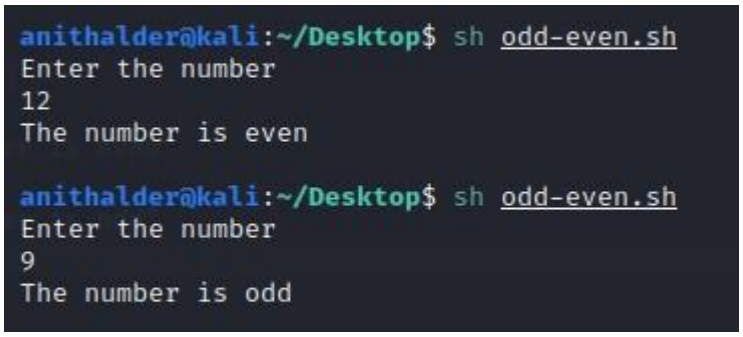
Steps :

```
1. Start
2. If (num % 2 = 0) then    //num is the variable which hold value
   2.1. Print "The number is even"
3. Else
   3.1. Print "The number is odd"
4. End if
5. Stop
```

Sorce code :

```
echo "Enter the number"
read n
s=`expr $n % 2`
if [ $s -eq 0 ]
then
echo "The number is even"
else
echo "The number is odd"
fi
```

Output :



```
anithalder@kali:~/Desktop$ sh odd-even.sh
Enter the number
12
The number is even

anithalder@kali:~/Desktop$ sh odd-even.sh
Enter the number
9
The number is odd
```

9. Write a shell script to check wheather a year is leapyear or not.

Agorithm Leapyear

Input : Read the year from the user.

Output : Print the year is leapyear or not

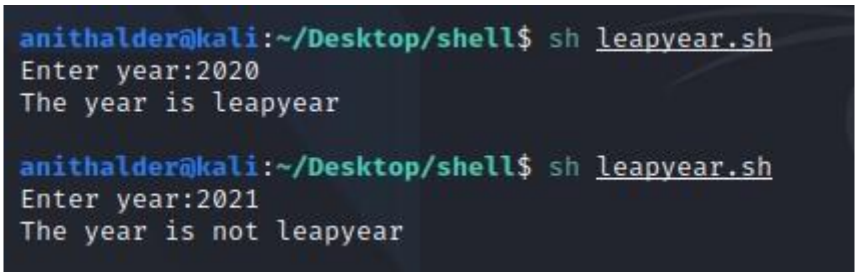
Steps :

```
1. Start
2. If (year % 400 = 0 OR year % 100 = 0  AND  year % 4 = 0)then
   2.1. Print  "The year is leap year"
3. Else
   3.1. Print "The year is not leap year"
4. End if
5. Stop
```

Sorce code :

```
read -p "Enter year:" year
if [ `expr $year % 400` -eq 0 -a `expr $year % 100` -eq 0 -o `expr
$year % 4` -eq 0 ]
then
        echo "The year is leapyear"
        else
        echo "The year is not leapyear"
fi
```

Output :



```
anithalder@kali:~/Desktop/shell$ sh leapyear.sh
Enter year:2020
The year is leapyear

anithalder@kali:~/Desktop/shell$ sh leapyear.sh
Enter year:2021
The year is not leapyear
```

10. Write a shell script to print a name n number of times.

Algorithm Name_Print

Input : Read a name from the user and number of times.
Output : Print the name N number of times.

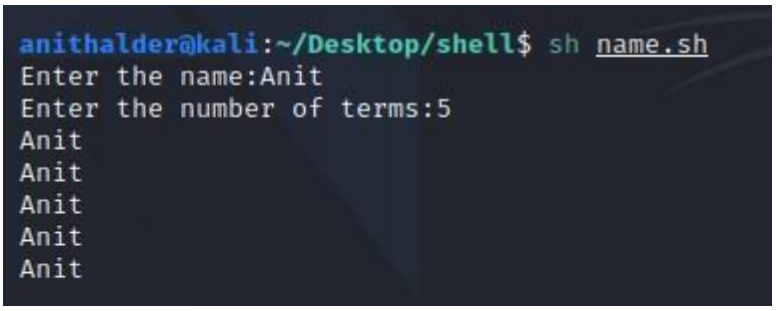
Steps :

```
1. Start
2. Name = "Anit Halder"
3. While ( I < N ) do
   3.1. Print  Name
4. End while
5. End
```

Sorce code :

```
read -p "Enter the name:" nam
read -p "Enter the number of terms:" n
i=0
while [ $i -lt $n ]
do
echo $nam
i=`expr $i + 1`
done
```

Output :



```
anithalder@kali:~/Desktop/shell$ sh name.sh
Enter the name:Anit
Enter the number of terms:5
Anit
Anit
Anit
Anit
Anit
```

11. Write a shell script to calculate the perimeter and area of geometric structures.

Algorithm Calculate_Areas&Perimeters_GeoShapes ()

Input : A choice is taken as given input from the user , then the required for shapes are also given

Output : The Perimeters and Areas of a specific geometrical shapes are displayed on the screen

Steps :

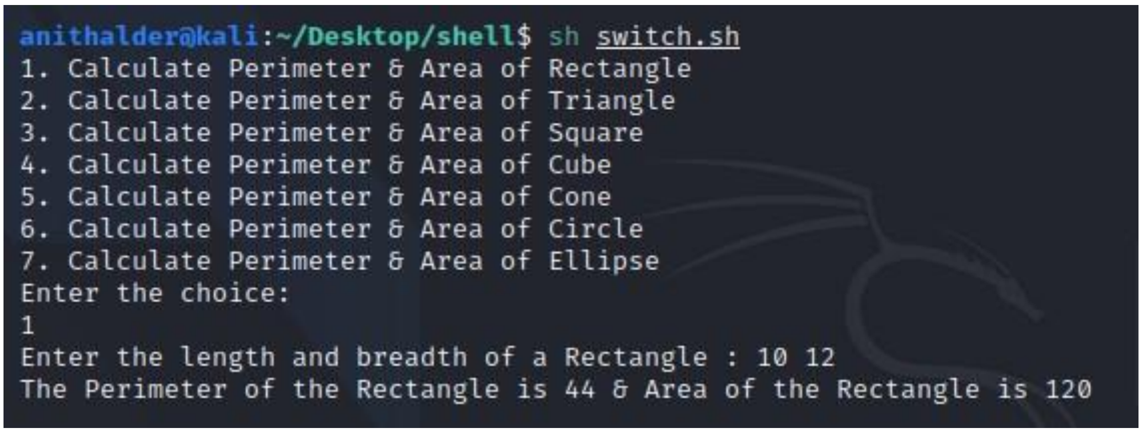
```
1. Start
2. Print "1. Calculate Perimeter and Area of Rectangle"
3. Print "2. Calculate Perimeter and Area of Triangle"
4. Print "3. Calculate Perimeter and Area of Square"
5. Print "4. Calculate Perimeter and Area of Cube"
6. Print "5. Calculate Perimeter and Area of Cone"
7. Print "6. Calculate Perimeter and Area of Circle"
8. Print "7. Calculate Perimeter and Area of Ellipse"
9. Read choice
10. case 1:
    10.1. Read length , breadth of Rectangle
    10.2. P1 <— 2 * (length + breadth)
    10.3. A1 <— length * breadth
    10.4. Print "The Perimeter of the Rectangle is", P1 ,"& Area of the Rectangle is" A1
    10.5. Break
11. case 2:
    11.1. Read side1 , side2 , side3 of Triangle
    11.2. P2 <— side1 + side2 + side3
    11.3. S <— P2 / 2
    11.4. A2 <— sqrt ( S * (S – side1) * (S – side2) * (S – side3) )
    11.5. Print "The Perimeter of the Triangle is", P2 ,"& Area of the Triangle is" A2
    11.6. Break
12. case 3:
    12.1. Read side of Square
    12.2. P3 <— 4 * side
    12.3. A3 <— side ^ 2
    12.4. Print "The Perimeter of the Square is", P3 ,"& Area of the Square is" A3
    12.5. Break
13. case 4:
    13.1. Read SIDE of Cube
    13.2. P4 <— 12 * SIDE
    13.3. A4 <— 6 * (SIDE ^ 2)
    13.4. Print "The Perimeter of the Cube is", P4 ,"& Area of the Cube is" A4
    13.5. Break
14. case 5:
    14.1. Read radius , l
    14.2. P5 <— 2 * (22 / 7) * radius
    14.3. A5 <— (22 / 7) * radius * (radius + l)
    14.4. Print "The Perimeter of the Cone is", P5 ,"& Area of the Cone is" A5
    14.5. Break
15. case 6:
    15.1. Read radius1
    15.2. P6 <— 2 * (22 / 7) * radius1
    15.3. A6 <— (22 / 7) * (radius1 ^ 2)
    15.4. Print "The Perimeter of the Circle is", P6 ,"& Area of the Circle is" A6
    15.5. Break
16. case 7:
    16.1. Read major , minor
    16.2. P7 <— [ ( 3 * (22 / 7) * ((major - minor) ^ 2) ) / ( sqrt ((major^2) + (14 * major * minor) + (minor^2)) + (10 * (major + minor)) ) ] + [ (22 / 7) * (major + minor) ]
    16.3. A7 <— (22 / 7) * major * minor
    16.4. Print "The Perimeter of the Ellipse is", P7 ,"& Area of the Ellipse is" A7
    16.5. Break
17. default:
    17.1. Print "Please run the program again and Seeing the options, Choose the correct choice... "
18. End
```

Sorce code :

```
echo "1. Calculate Perimeter & Area of Rectangle"
echo "2. Calculate Perimeter & Area of Triangle"
echo "3. Calculate Perimeter & Area of Square"
echo "4. Calculate Perimeter & Area of Cube"
echo "5. Calculate Perimeter & Area of Cone"
echo "6. Calculate Perimeter & Area of Circle"
echo "7. Calculate Perimeter & Area of Ellipse"

echo "Enter the choice: "
read choice

case $choice in

1)      echo -n "Enter the length and breadth of a Rectangle : "
```

```
        read length1 breadth1
        P1=`echo "scale=2 ; 2 * ($length1 + $breadth1)" | bc`
        A1=`echo "scale=2 ; ($length1 * $breadth1)" | bc`
        echo "The Perimeter of the Rectangle is" $P1 "& Area of the
Rectangle is" $A1;;

2)      echo -n "Enter the three sides : "
        read side1 side2 side3
        P2=`echo "scale=2 ; ($side1 + $side2 + $side3)" | bc`
        S=`echo "scale=2 ; ($P2 / 2)" | bc`
        A2=`echo "scale=2 ; sqrt( $S * ($S - $side1) * ($S - $side2)
* ($S - $side3) )" | bc`
        echo "The Perimeter of the Triangle is" $P2 "& Area of the
Triangle is" $A2;;

3)      echo -n "Enter the Side of Square : "
        read side
        P3=`echo "scale=2 ; (4 * $side)" | bc`
        A3=`echo "scale=2 ; ($side ^ 2)" | bc`
        echo "The Perimeter of the Square is" $P3 "& Area pf the
Square is" $A3;;

4)      echo -n "Enter the Side of Cube : "
        read SIDE
        P4=`echo "scale=2 ; (12 * $SIDE)" | bc`
        A4=`echo "scale=2 ; 6 * ($SIDE ^ 2)" | bc`
        echo "The Perimeter of the Cube is" $P4 "& Area of the
Cube is" $A4;;

5)      echo -n "Enter of the Radius of the Cone : "
        read radius
        echo -n "Enter the Slant Height of the Cone : "
        read l
        P5=`echo "scale=2 ; 2 * (22 / 7) * $radius" | bc`
        A5=`echo "scale=2 ; (22 / 7) * $radius * ($radius + $l)" |
bc`
        echo "The Perimeter of the Cone is" $P5 "& Area of the
Cone is" $A5;;

6)      echo -n "Enter of the Radius of the Circle : "
        read radius1
        P6=`echo "scale=2 ; 2 * (22 / 7) * $radius1" | bc`
        A6=`echo "scale=2 ; (22 / 7) * ($radius1 ^ 2)" | bc`
        echo "The Perimeter of the Circle is" $P6 "& Area of the
Circle is" $A6;;

7)      echo "Enter the Radius of Major Axis & Minor Axis : "
        read major minor
        P7=`echo "scale=4 ; (3 * (22/ 7) * (($major - $minor) ^ 2) / ( sqrt(($major ^ 2) + (14 * $major * $minor) + ($minor ^ 2)) + (10 * ($major + $minor)) )) + ((22 / 7) * ($major + $minor))" | bc`
        A7=`echo "scale=2 ; (22 / 7) * $major * $minor" | bc`
        echo "The Perimeter of the Ellipse is" $P7 "& Area of the Ellipse is" $A7;;

*)      echo "Please run the programme again & seeing the options, Choose the correct Choice...";;
esac
```

Output :



```
anithalder@kali:~/Desktop/shell$ sh switch.sh
1. Calculate Perimeter & Area of Rectangle
2. Calculate Perimeter & Area of Triangle
3. Calculate Perimeter & Area of Square
4. Calculate Perimeter & Area of Cube
5. Calculate Perimeter & Area of Cone
6. Calculate Perimeter & Area of Circle
7. Calculate Perimeter & Area of Ellipse
Enter the choice:
1
Enter the length and breadth of a Rectangle : 10 12
The Perimeter of the Rectangle is 44 & Area of the Rectangle is 120
```

12. Write a shell script to determine person is eligible to vote or not. If B or C are not eligble display how many years left to eligible.

Algorithm Voter_Elegibility

Input : Read the year of the persons birth.

Output : Print the year is elegilble or not and also print if the person is not eligible how many year left to eligible.

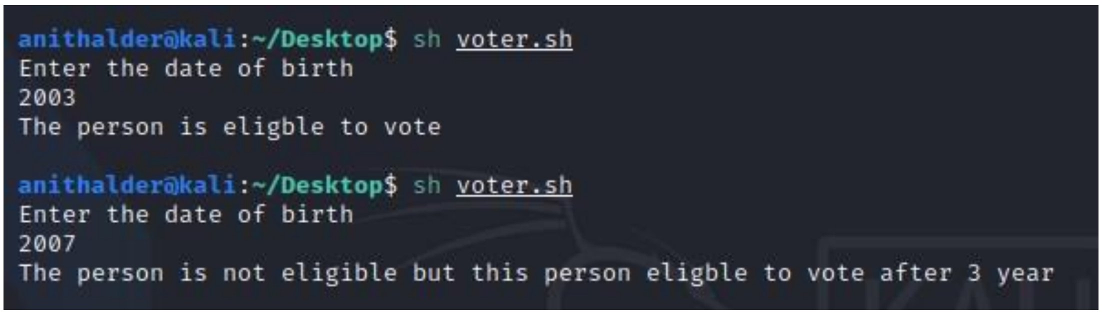
Steps :

```
1.  Start
2.  Year ← X
3.  Now ← latest year
4.  Dif ← Now - X
5.  If (year >= 18 )
        3.1. Print "Eligible for vote"
6.  Else
        4.1. Print "Not eligible for vote but this person eligible after"
             Dif "year"
7.  End if
8.  End
```

Sorce code :

```
echo "Enter the date of birth"
read birth
year=`date +%"Y"`
age=`expr $year - $birth`
if [ $age -gt 18 ]
then
        echo "The person is eligble to vote"
else
        rem=`expr 18 - $age`
        echo "The person is not eligible but this person eligble to vote after "$rem" year"
fi
```

Output :



```
anithalder@kali:~/Desktop$ sh voter.sh
Enter the date of birth
2003
The person is eligble to vote

anithalder@kali:~/Desktop$ sh voter.sh
Enter the date of birth
2007
The person is not eligible but this person eligble to vote after 3 year
```

14. Write shell script to show the all natural numbers fom 1 to n (n is taken from the user).

Algorithm Natural_numbers_print :

Input : The limit of the series is taken from the user and store it in N.

Output : Print the series of the 1 to Nnatural numbers.

Steps :

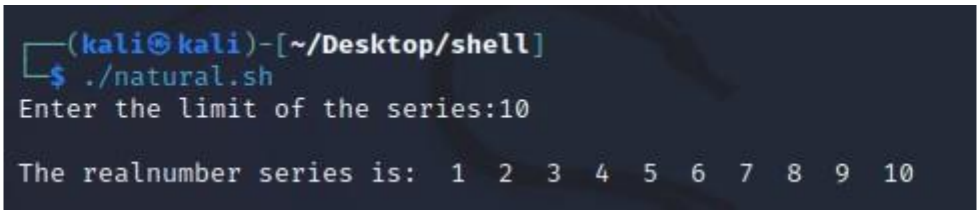
```
1. Start
2. I ← 1               //Hold the N terms of natural numbers
3. For (I to N) do     //N is the limit of the series given by the user
   3.1.   Print I
4. End for
5. End
```

Discussion : Here we print the all natural number from 1 to N and the Nis given by the user I is the number of iteration and the natural numbers.

Source code :

```
read -p "Enter the limit of the series:" n
i=1
echo -n "\nThe realnumber series is:"
while [ $i -le $n ]
do
        echo -n " " $i
        i=`expr $i + 1`
done
```

Output :



```
┌──(kali㉿kali)-[~/Desktop/shell]
└─$ ./natural.sh
Enter the limit of the series:10

The realnumber series is:  1  2  3  4  5  6  7  8  9  10
```

15. Display all even & odd number from 1 to N.(Where N is given by the user).

Algorithm Even_Odd_series :

Input : Enter the limit of the series and store it to N. And N is given by the user.

Output : Print the series of the even numbers & the series of the odd numbers from 1 to N.

Steps :

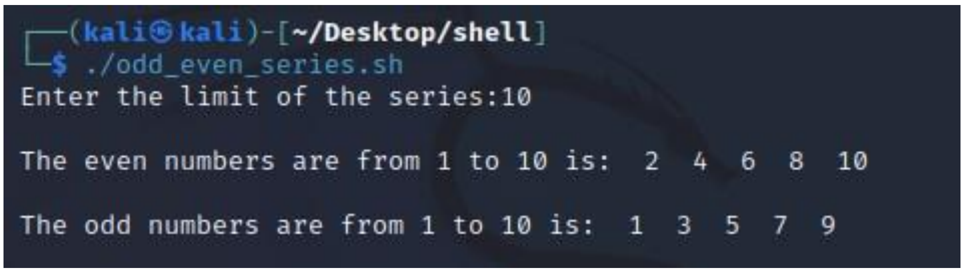
```
1. Start
2. I←1
3. Print "The even numbers are:"
4. For ( I ← 1 to N ) do
   4.1.  If ( I % 2 = 0 ) then
      4.1.1. Print  I
   4.2.  End if
5. End for
6. I←1
7. Print "The odd numbers are:"
8. For ( I ← 1 to N ) do
   8.1.  If ( I % 2 ≠ 0 ) then
      8.1.1. Print  I
   8.2.  End if
9. End for
10. End
```

Souce code :

```
read -p "Enter the limit of the series:" n
i=1
echo -n "\n""The even numbers are from "$i" to "$n" is:"
while [ $i -le $n ]
do
        if [ `expr $i % 2` -eq 0 ]
        then
        echo -n " " $i
        fi
i=`expr $i + 1`
done
i=1
echo -n "\n\n""The odd numbers are from "$i" to "$n" is:"
```

```
while [ $i -le $n ]
do
        if [ `expr $i % 2` -ne 0 ]
        then
        echo -n " " $i
        fi
i=`expr $i + 1`
done
```

Output :



```
┌──(kali㉿kali)-[~/Desktop/shell]
└─$ ./odd_even_series.sh
Enter the limit of the series:10

The even numbers are from 1 to 10 is:  2  4  6  8  10

The odd numbers are from 1 to 10 is:  1  3  5  7  9
```

16. Write a shell script program to display all the factors of the user given number.

Algorithm Factors_of_number :

Input : Enter the number to calculate the factors you want . And store it into a Variable N.

Output : Print the all factors of the N.

Steps :

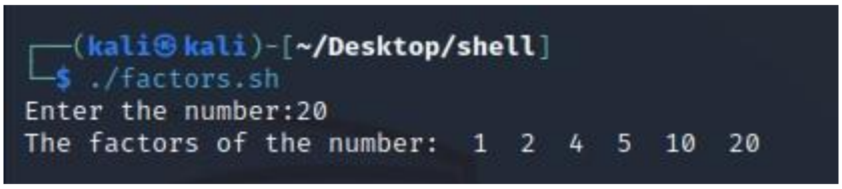
```
1. Start
2. I ← 1
3. For (I to N) do
   10.1. If ( ( N % I ) = 0 ) then
      10.1.1. Print I
   10.2. End if
4. End for
5. End
```

Discussion : Here we print the all factors of user given number and we increase I and devide it with N and check the remainder value is zero or not if the the remainder value is zero then the value of the **I** and declear it as factor of N.

Source code :

```
read -p "Enter the number:" val
echo -n "The factors of the number:"
i=1
while [ $i -le $val ]
do
        if [ `expr $val % $i` -eq 0 ]
        then
        echo -n " " $i
        fi
i=`expr $i + 1`
done
```

Output :



```
┌──(kali㉿kali)-[~/Desktop/shell]
└─$ ./factors.sh
Enter the number:20
The factors of the number:  1  2  4  5  10  20
```

17. Write a shell script program to check weather a number is perfect or not.

Algorithm Perfect_or_Not :

Input : N is the number which is given by the user to check weather number is perfect or not.

Output : Print the number is perfect or not.

Steps :

```
1. Start
2. I ← 1 , SUM ← 0
3. For [I to (N / 2)] do
        3.1.    If [ ( N % I ) = 0 ] then
        3.1.1.  SUM ← SUM + I
        3.2.    End if
4. End for
5. If (SUM = N) then
        5.1.    Print “The number is perfect”
        5.2.    Else
        5.2.1.  Print “The number is not perfect”
6. End if
7. End
```

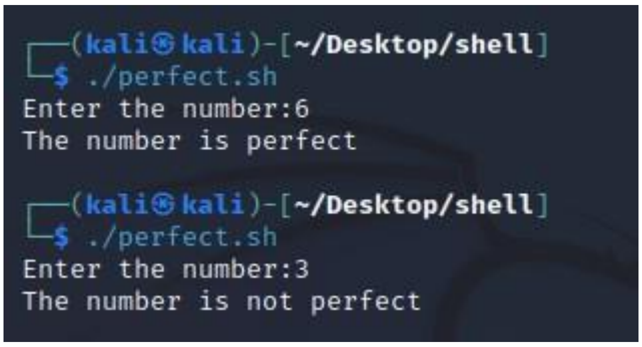
Discussion : Here the SUM is the variable where the summation of all factors of a number is store and N is the number which is given by the user and I is the iteration number and we find of that numbers all factors and add it to find the number is perfect or not.

Source code :

```
read -p "Enter the number:" n
i=1
sum=0
while [ $i -le `expr $n / 2` ]
do
        if [ `expr $n % $i` -eq 0 ]
        then
        sum=`expr $sum + $i`
        fi
i=`expr $i + 1`
done
```

```
if [ $sum -eq $n ]
then
echo "The number is perfect"
else
echo "The number is not perfect"
fi
```

Output :



```
┌──(kali㉿kali)-[~/Desktop/shell]
└─$ ./perfect.sh
Enter the number:6
The number is perfect

┌──(kali㉿kali)-[~/Desktop/shell]
└─$ ./perfect.sh
Enter the number:3
The number is not perfect
```

18. Write a shell script to print the factorial of a user given number.

Algorithm Factorial :

Input : N is the number which is given by the user.
Output : Print the facotorial of the number.

<u>Steps</u> :

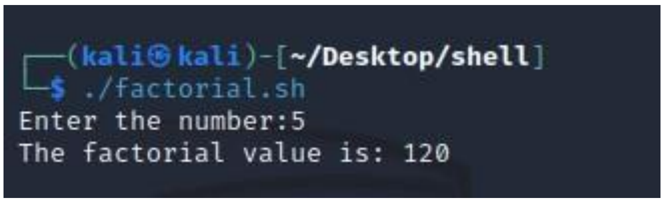
```
1. Start
2. I ← 1 , F ← 1
3. For ( I to N ) do
       3.1.   F ← F * I
4. End for
5. Print F
6. End
```

Discussion : Here the F is the variable which holds the facor value of the guven number and N is the number which is given by the user and I is the iteration number.

Source code :

```
read -p "Enter the number:" num
fact=1
i=1
while [ $i -le $num ]
do
        fact=`expr $fact \* $i`
        i=`expr $i + 1`
done
echo "The factorial value is:" $fact
```

Output :



```
┌──(kali㉿kali)-[~/Desktop/shell]
└─$ ./factorial.sh
Enter the number:5
The factorial value is: 120
```

19. Write a shell script to print this following series sum :
1 ! + 2 ! + 3 ! +…………….+ N ! [Here N is the user given number].

Algorithm Factorial_Series :

Input : N is the number which is given by the user.
Output : Print the facotorial series up to N.

<u>Steps</u> :

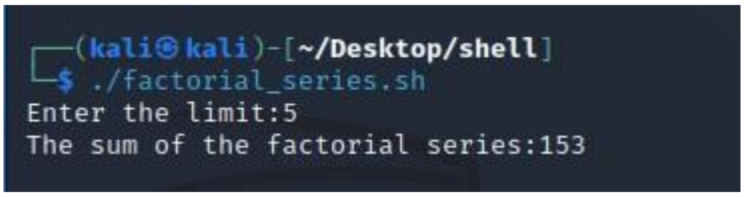
```
1. Start
2. I ← 1 , F ← 1 ,J ← 1 , SUM ← 0
3. For ( J to N ) do
        3.1.    I ← 1 , F ← 1
        3.2.    For ( I to J ) do
                3.2.1.  F ← F * I
        3.3.    End for
        3.4.    SUM ← SUM + F
4. End for
5. Print SUM
6. End
```

Discussion : Here the F is the variable which holds the facor value of the series and N is the limit of the series which is given by the user and I is the iteration number & SUM is the addition of the series.

Source code :

```
read -p "Enter the limit:" num
sum=0
j=1
while [ $j -le $num ]
do
        i=1
        fact=1
        while [ $i -le $j ]
        do
                fact=`expr $fact \* $i`
                i=`expr $i + 1`
        done
        sum=`expr $sum + $fact`
j=`expr $j + 1`
done
echo -n "The sum of the factorial series:"$sum
```

Output :



```
┌──(kali㉿kali)-[~/Desktop/shell]
└─$ ./factorial_series.sh
Enter the limit:5
The sum of the factorial series:153
```

20. Write a shell script to display this series : 1 , 4 , 7 , 10 , 13 , ………… , N th term.

Algorithm Series_3 :

Input : N is the number of terms which is given by the user.
Output : Print the series.

Steps :

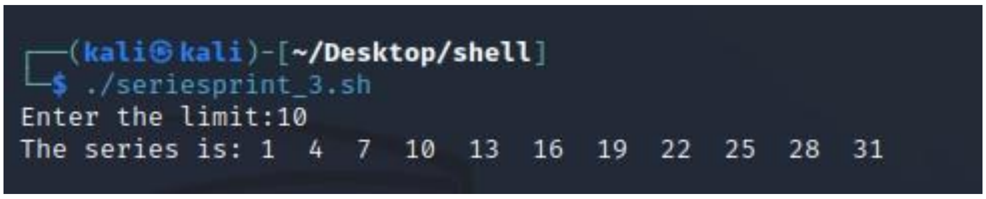
```
1. Start
2. I ← 1 , J ← 1
3. Print I
4. For ( I to N ) do
        4.1.   J ← J + 3
        4.2.   Print J
5. End for
6. End
```

Discussion : Here the N is the number of terms and I is the iteration number and J is the another iteration number which which gives this series an another rhythm and print the series .

Source code :

```
read -p "Enter the limit:" num
i=1
j=1
echo -n "The series is:" $i
while [ $j -le $num ]
do
        i=`expr $i + 3`
        echo -n " " $i
j=`expr $j + 1`
done
```

Output :



```
┌──(kali㉿kali)-[~/Desktop/shell]
└─$ ./seriesprint_3.sh
Enter the limit:10
The series is: 1  4  7  10  13  16  19  22  25  28  31
```

21. Write a shell scrip to print the fibonaci series : 0 , 1 , 1 , 2 , 3 , 5 , ………… , N^th term.

Algorithm Fibonaci :

Input : N is the number of terms which is given by the user.
Output : Print the series.

__Steps__ :

```
1. Start
2. I ← 0 , A ← 0 , B ← 1
3. Print A,B
4. For ( I ← 0 to N - 2 ) do
    4.1.  C ← A + B
    4.2.  A ← B
    4.3.  B ← C
    4.4.  Print C
5. End for
6. End
```

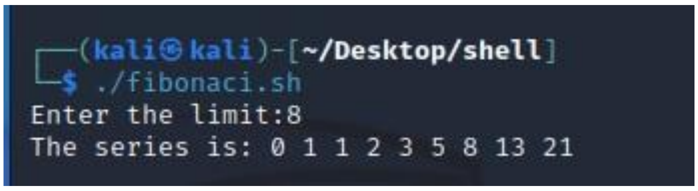
Discussion : Here the N is the number of terms and I is the iteration number and A & B is variable which holds the previous two numbers C holds the addition of the previous numbers and print the series.

Source code :

```
read -p "Enter the limit:" f
i=0
a=0
b=1
echo -n "The series is:" $a $b
while [ $i -le `expr $f - 2` ]
do
        c=`expr $a + $b`
        a=$b
```

```
            b=$c
            echo -n " "$c
    i=`expr $i + 1`
    done
```

Output :



```
┌──(kali㉿kali)-[~/Desktop/shell]
└─$ ./fibonaci.sh
Enter the limit:8
The series is: 0 1 1 2 3 5 8 13 21
```

22. Write a shell script to print the GCD & LCM of two numbers.

Algorithm GCD_LCM :

Input : N_1 , N_2 is given by the user .
Output : Print the GCD & LCM of the two numbers.

<u>Steps</u> :

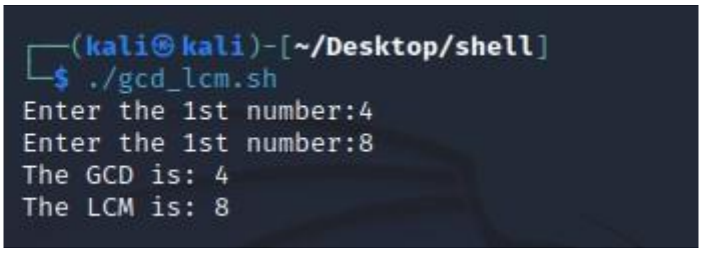
```
1. Start
2. I ← 0
3. If ( N₁ < N₂ )
    3.1.   NUM ← N₂
    3.2.   DEN ← N₁
4. Else
    4.1.   NUM ← N₁
    4.2.   DEN ← N₂
5. End if
6. REM ← ( NUM % DEN )
7. While ( REM ≠ 0 ) do
    7.1.   NUM ← DEN
    7.2.   DEN ← REM
    7.3.   REM ← NUM % DEN
8. End While
9. GCD ← DEN
10. LCM ← ( N₁ * N₂ ) / GCD
11. Print GCD , LCM
12. End
```

Discussion : Here the N₁ , N₂ is the two numbers given by the user and NUM & DEN is the two variable which means numerator & denominator and here GCD & LCM are the two variable which holds the result of the two numbers gcd and lcm values.

Source code :

```
read -p "Enter the 1st number:" n1
read -p "Enter the 1st number:" n2
if [ $n1 -gt $n2 ]
then
num=$n1
den=$n2
else
num=$n2
den=$n1
fi
rem=`expr $num % $den`
while [ $rem -ne 0 ]
do
        num=$den
        den=$rem
rem=`expr $num % $den`
done
gcd=$den
lcm=`expr $n1 \* $n2 / $gcd`
echo "The GCD is:" $gcd
echo "The LCM is:" $lcm
```

Output :



```
┌──(kali㉿kali)-[~/Desktop/shell]
└─$ ./gcd_lcm.sh
Enter the 1st number:4
Enter the 1st number:8
The GCD is: 4
The LCM is: 8
```

23. Write a shell script to check a number is amstrong or not.

Algorithm Amstrong :

Input : N is given by the user .
Output : Print the value is amstrong or not.

Steps :

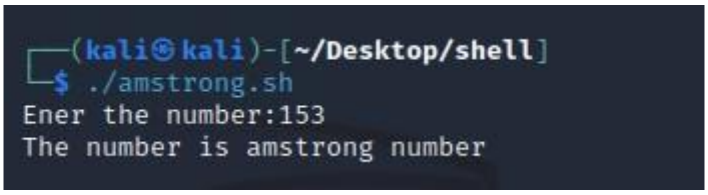
```
1. Start
2. SUM ← 0
3. TEMP ← N
4. While ( N > 0 ) do
      4.1.   R ← (TEMP % 10)
      4.2.   TEMP ← (TEMP / 10)
      4.3.   SUM ← SUM + (R * R * R)
5. End while
6. If (SUM = NUM) then
      6.1.   Print "The number is amstrong number"
7. Else
      7.1.   Print "The number is not amstrong number"
8. End
```

Discussion : Here the SUM is the variable which holds the addition of the numbers and TEMP is a temporary variable which holds the user given value an use to modify and N is the variable which holds the user given value .

Source code :

```
read -p "Ener the number:" num
temp=$num
sum=0
while [ $temp -gt 0 ]
do
        r=`expr $temp % 10`
        temp1=`expr $r \* $r \* $r`
        sum=`expr $sum + $temp1`
        temp=`expr $temp / 10`

done
if [ $sum -eq $num ]
then
        echo "The number is amstrong number"
else
        echo "The number is not amstrong number"
fi
```

Output :



```
┌──(kali㉿kali)-[~/Desktop/shell]
└─$ ./amstrong.sh
Ener the number:153
The number is amstrong number
```

24. Write shell script to check the number is prime or not.

Algorithm prime_not :

Input : N is taken by the user .
Output : Print the value is prime or not.

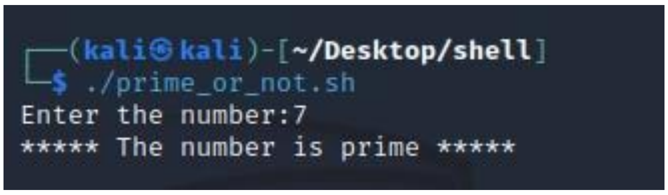
Steps :

```
1. Start
2. Read number
3. i ← 2
4. count ← 0
5. While i <= (number/2) Do
   5.1.  If (number%i) = 0 Then
         5.1.1. count ← count + 1
         5.1.2. Break
   5.2.  End If
   5.3.  i ← i + 1
6. End While
7. If count = 0 Then
   7.1.  Print "The number", number , "is a Prime Number"
8. Else
   8.1.  Print "The number", number , "is a Non-Prime Number"
9. End If
10. End
```

Source code :

```
read -p "Enter the number:" num
i=1
c=0
while [ $i -le $num ]
do
        if [ `expr $num % $i` -eq 0 ]
        then
        c=`expr $c + 1`
        fi
i=`expr $i + 1`
done
if [ $c -eq 2 ]
then
#echo "\n"
echo "***** The number is prime *****"
else
#echo "\n"
echo "***** The number is not prime *****"
fi read -p "Enter the number:" num
i=1
c=0
while [ $i -le $num ]
do
        if [ `expr $num % $i` -eq 0 ]
        then
        c=`expr $c + 1`
        fi
i=`expr $i + 1`
done
if [ $c -eq 2 ]
then
#echo "\n"
echo "***** The number is prime *****"
else
#echo "\n"
echo "***** The number is not prime *****"
fi
```

Output :



```
┌──(kali㉿kali)-[~/Desktop/shell]
└─$ ./prime_or_not.sh
Enter the number:7
***** The number is prime *****
```

25. Write a shell script to print this pattern :

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

Algorithm pattern :

Input : N is number of line that is given by the user.
Output : Print the pattern.

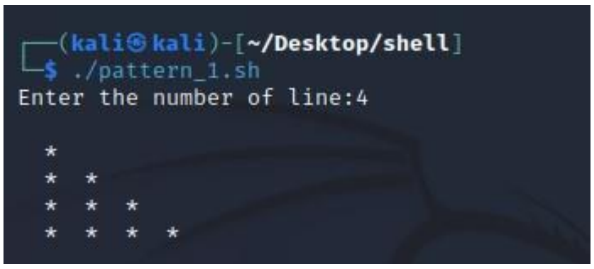
Steps :

```
1. Start
2. For ( I ← 0 to N ) do
    2.1.  J = 1
    2.2.  For ( J ← 0 to I )
        2.2.1. Print "*"
    2.3.  End for
    2.4.  Go to next line
3. End for
4. End
```

Source code :

```
read -p "Enter the number of line:" n
i=1
j=1
echo ""
while [ $i -le $n ]
do
        j=1
        while [ $j -le $i ]
        do
        echo -n "  *"
        j=`expr $j + 1`
        done
echo ""
i=`expr $i + 1`
done
```

Output :



```
┌──(kali㉿kali)-[~/Desktop/shell]
└─$ ./pattern_1.sh
Enter the number of line:4

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

26. Write a shell script to display all prime numbers from 1 to N.

Algorithm prime_series :

Input : N is taken by the user .

Output : Print the all prime numbers which belongs to 1 to N.

<u>Steps</u> :

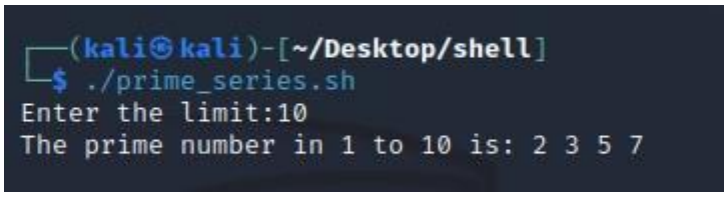
```
1. Start
2. Print "The prime numbers are:"
3. For ( I ← 1 to N ) do
   3.1.    C = 0 , J = 2
   3.2.    While ( J <= √I ) do
      3.2.1.  If ( I % J = 0 ) then
         3.2.1.1.    C ← C+1
         3.2.1.2.    Exit
      3.2.2.  End if
      3.2.3.  J ← J+1
   3.3.    End while
   3.4.    If ( C = 0 and I ≠ 1 ) then
      3.4.1.  Print I
4. End for
5. End
```

Source code :

```
read -p "Enter the limit:" n
i=1
echo -n "The prime number in "$i" to "$n" is:"
while [ $i -le $n ]
do
        c=0
        j=2
        square=`echo "sqrt($i)" | bc`
        while [ $j -le $square ]
        do
                if [ `expr $i % $j` -eq 0 ]
                then
                c=`expr $c + 1`
                break
                fi
        j=`expr $j + 1`
```

```
                done
                if [ $c -eq 0 -a $i -ne 1 ]
                then
                echo -n " "$i
                fi
        i=`expr $i + 1`
        done
```

Output :



```
┌──(kali㉿kali)-[~/Desktop/shell]
└─$ ./prime_series.sh
Enter the limit:10
The prime number in 1 to 10 is: 2 3 5 7
```

27. Write a shell script to print this pattern :

1
1 2
1 2 3
1 2 3 4

Algorithm pattern :

Input : N is number of line that is given by the user.
Output : Print the pattern.

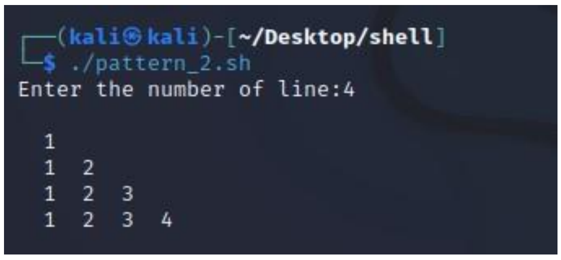
Steps :

```
1. Start
2. For ( I ← 0 to N ) do
    2.1. For ( J ← 0 to I )
        2.1.1. Print J
    2.2. End for
    2.3. Go to next line
3. End for
4. End
```

Source code :

```
read -p "Enter the number of line:" n
i=1
j=1
echo ""
while [ $i -le $n ]
do
        j=1
        while [ $j -le $i ]
        do
        echo -n "  "$j
        j=`expr $j + 1`
        done
echo ""
i=`expr $i + 1`
done
```

Output :



```
┌──(kali㉿kali)-[~/Desktop/shell]
└─$ ./pattern_2.sh
Enter the number of line:4

  1
  1  2
  1  2  3
  1  2  3  4
```

28. Write a shell script to connect the five programs using switch case.

Algorithm switch_5 :

Input : Enter your choise .
Output : Print the output to the corresponding choice .

Steps :

```
1. Start
2. While ( 1 )
   2.1.  Print "1. For calculator"
   2.2.  Print "2. For leapyear or not"
   2.3.  Print "3. For Eligible for voter or not"
   2.4.  Print "4. For even or odd"
   2.5.  Print "5. For prime or not prime"
   2.6.  Print "6. For exit press 0"
   2.7.  Print "Enter your choice:"
   2.8.  Read choice
   2.9.  Case 1:
         2.9.1.  Print "1. For sum (+)"
         2.9.2.  Print "2. For substraction (-)"
         2.9.3.  Print "3. For multiplication (*)"
         2.9.4.  Print "4. For division (/)"
         2.9.5.  Print "5. For exit press 0"
         2.9.6.  Print "Enter your choice:"
         2.9.7.  Read choice
         2.9.8.  Case 1:
                 2.9.8.1.   Read N1
                 2.9.8.2.   Read N2
                 2.9.8.3.   SUM ← N1 + N2
                 2.9.8.4.   Print SUM
                 2.9.8.5.   Break
         2.9.9.  Case 2:
                 2.9.9.1.   Read N1
                 2.9.9.2.   Read N2
                 2.9.9.3.   SUB ← N1 - N2
                 2.9.9.4.   Print SUB
                 2.9.9.5.   Break
         2.9.10. Case 3:
                 2.9.10.1.  Read N1
```

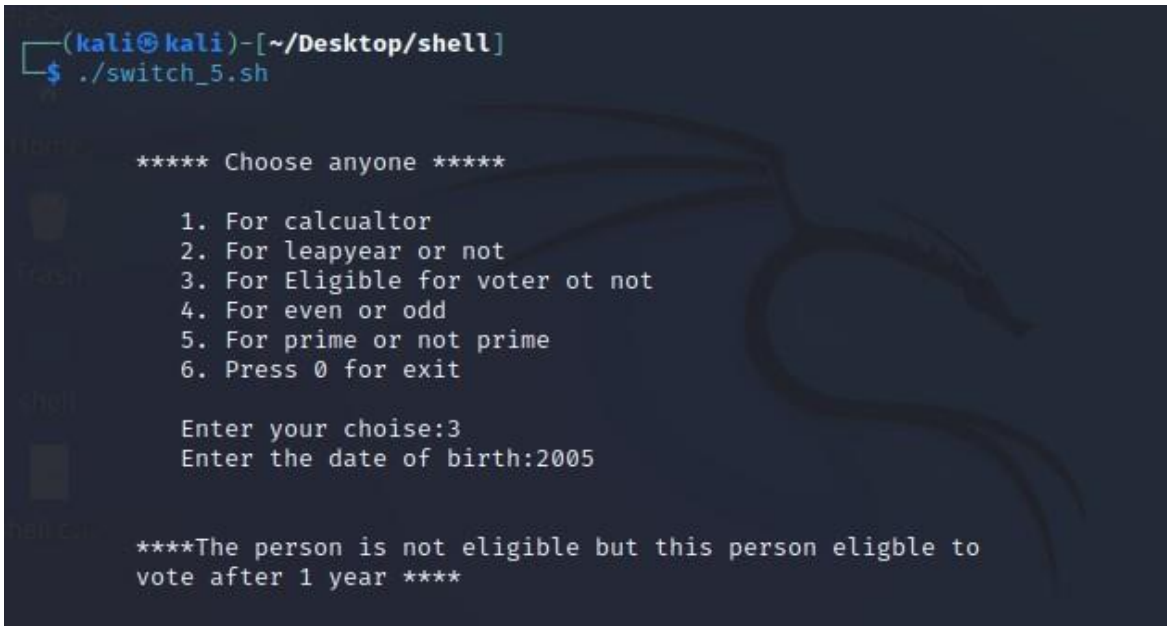
```
                2.9.10.2. Read N2
                2.9.10.3. MUL ← N1 * N2
                2.9.10.4. Print MUL
                2.9.10.5. Break
        2.9.11. Case 4:
                2.9.11.1. Read N1
                2.9.11.2. Read N2
                2.9.11.3. DIV ← N1 / N2
                2.9.11.4. Print DIV
                2.9.11.5. Break
        2.9.12. Case 0:
                2.9.12.1. Exit
        2.9.13. Default :
                2.9.13.1. Print "! ! ! ! Please choose in
                          given option ! ! ! !"
        2.9.14. End
    2.10. Case 2:
        2.10.1. Algorithm Leapyear ( )
        2.10.2. Break
    2.11. Case 3:
        2.11.1. Algorithm Voter_Elegibility ( )
        2.11.2. Break
    2.12. Case 4:
        2.12.1. Algorithm even_odd ( )
        2.12.2. Break
    2.13. Case 5:
        2.13.1. Algorithm prime_or_not ( )
        2.13.2. Break
    2.14. Case 6:
        2.14.1. Exit
    2.15. Default :
        2.15.1. Print "! ! ! Please choose in given option ! ! !"
3. End while
4. End
```

Source code :

```
while [ 1 ]
do
echo "\n"
echo "\t* * * * * Choose anyone * * * * *""\n"
echo "\t   1. For calcualtor"
echo "\t   2. For leapyear or not"
echo "\t   3. For Eligible for voter ot not"
echo "\t   4. For even or odd"
echo "\t   5. For prime or not prime"
echo "\t   6. Press 0 for exit""\n"
echo -n "\t   Enter your choise:"
read ch
case $ch in
        1)while [ 1 ]
          do
                echo "\n"
                echo "\t***** Choose any one in calculator *****""\n"
                echo "\t    1. For sum (+)"
                echo "\t    2. For substraction (-)"
                echo "\t    3. For multiplication (*)"
                echo "\t    4. For division (/)"
                echo "\t    5. Press 0 for exit calculator""\n"
                echo -n "\t   Enter your choise:"
                read ch
                echo "\n"
                case $ch in
                        1)echo -n "\t   Enter the 1st number:"
                          read n1
                          echo -n "\t   Enter the 2nd number:"
                          read n2
                          sum=`expr $n1 + $n2`
                          echo "\t   The sum of the thease numbers is:"
$sum;;

                        2)echo -n "\t   Enter the 1st number:"
                          read n1
                          echo -n "\t   Enter the 2nd number:"
                          read n2
                          sub=`expr $n1 - $n2`
                          echo "\t   The substraction of the thease numbers
is:" $sub;;

                        3)echo -n "\t   Enter the 1st number:"
                          read n1
                          echo -n "\t   Enter the 2nd number:"
                          read n2
                          mul=`expr $n1 \* $n2`
                          echo "\t   The multiplication of the thease numbers
is:" $mul;;

                        4)echo -n "\t   Enter the 1st number:"
                          read n1
                          echo -n "\t   Enter the 2nd number:"
                          read n2
                          div=`expr $n1 / $n2`
                          echo "\t   The division of the thease numbers
is:"$div;;

                        0)exit;;

                        *)echo "    ! ! ! Please choose in given option ! ! ! ";;
             esac
             done;;

           2)echo -n "\t   Enter year:"
             read year
             if [ `expr $year % 400` -eq 0 -a `expr $year % 100` -eq 0 -o
`expr $year % 4` -eq 0 ]
             then
             echo "\n\n""\t* * * * * The year is leapyear * * * * *"
           else
             echo "\n\n""\t* * * * * The year is not leapyear * * * * *"
             fi;;

           3)echo -n "\t   Enter the date of birth:"
           read birth
           year=`date +%"Y"`
           age=`expr $year - $birth`
           if [ $age -gt 18 ]
           then
           echo "\n\n""\t* * * * * The person is eligble to vote * * * * *"
           else
           rem=`expr 18 - $age`
           echo "\n\n""\t* * * *The person is not eligible but this person eligble
to
           vote after "$rem" year * * * *"
           fi;;

           4)echo -n "\t   Enter the number:"
           read n
           s=`expr $n % 2`
           if [ $s -eq 0 ]
           then
           echo "\n\n""\t* * * * *The number is even * * * * *"
           else
           echo "\n\n""\t* * * * *The number is odd * * * * *"
           fi;;

           5)echo -n "\t   Enter the number:"
           read num
           i=1
           c=0
```

```
                while [ $i -le $num ]
                do
                        if [ `expr $num % $i` -eq 0 ]
                        then
                        c=`expr $c + 1`
                        fi
                i=`expr $i + 1`
                done
                if [ $c -eq 2 ]
                then
                echo "\n\n""\t* * * * * The number is prime * * * * *"
                else
                echo "\n\n""\t* * * * * The number is not prime * * * * *"
                fi;;

                0)exit;;
                *)echo "    ! ! ! Please choose in given option ! ! ! ";;
        esac
        done
```

Output :



```
┌──(kali㉿kali)-[~/Desktop/shell]
└─$ ./switch_5.sh


      ***** Choose anyone *****

         1. For calcualtor
         2. For leapyear or not
         3. For Eligible for voter ot not
         4. For even or odd
         5. For prime or not prime
         6. Press 0 for exit

         Enter your choise:3
         Enter the date of birth:2005


      ****The person is not eligible but this person eligble to
      vote after 1 year ****
```