DAY 1

1.Generation of number series 1,2,3, 4, ……. N

ALGORITHM:

Step1: start the program.

Step2: Give input or limit say n.

Step3: Using for loop, check the condition

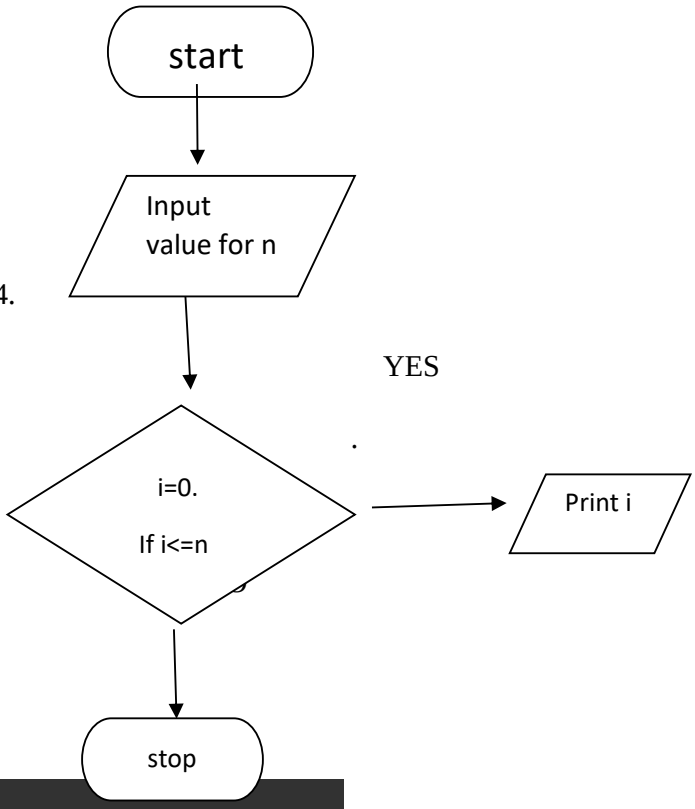
IF condition is TRUE, GO TO step 4.

ELSE, GO TO step 5.

Step4: print n

Step5: go to step3.

Step6: stop the program.




```c
#include <stdio.h>
int main()
{
    int i, n;
    printf("Enter any number: ");
    scanf("%d", &n);
    printf("Natural numbers from 1 to %d : \n", n);

    for(i=1; i<=n; i++)
    {
        printf("%d\n", i);
    }
    return 0;
 }
```

```
19
20
21
22
```

2.Summing up series 1+2+3+4…… +n

ALGORITHM:

Step1: start the program.

Step2: Give input n .

Step3: initialize sum=0.

Step4: use for loop,i<=n.

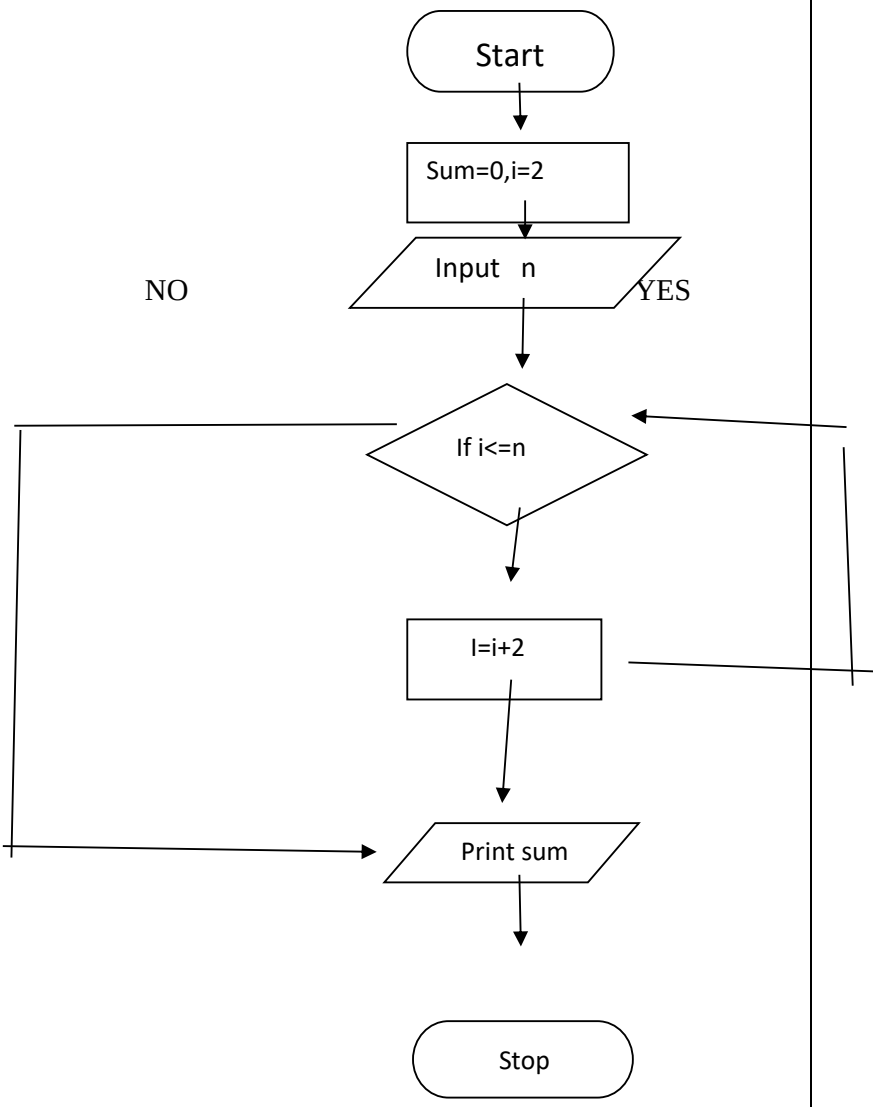
If true go to step 5.

Else go to step7 . NO YES

step5:sum=sum+i.

Step6: print sum.

Step7: stop




```c
#include<stdio.h>
int main()
{
   int n, sum=0;

   printf("Enter n value: ");
   scanf("%d",&n);

   for(int i=1; i<=n; i++)
   {
      printf("%d+",i);
      sum += i; //sum = sum + i;
   }

   printf("\b=%d",sum);

   return 0;
}
```

```
Enter n value: 2
1+2=3
```

3 Generation of even number series 2, 4, 6, …..n

ALGORITHM:

Step1: start the program.

Step2: Give input or limit say n.

Step3: Using for loop, check the condition

IF condition is TRUE, GO TO step 4.

ELSE, GO TO step 5.

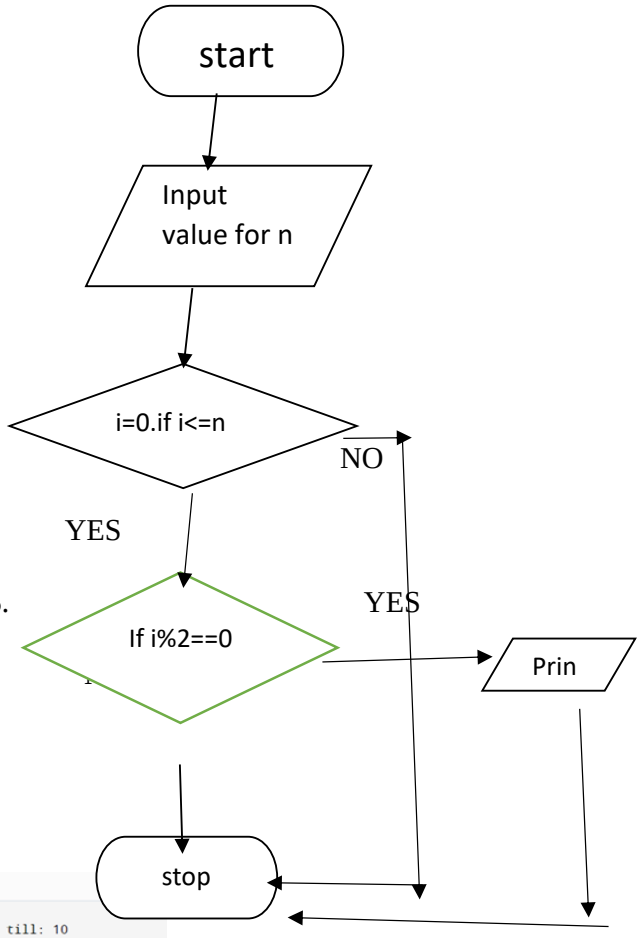
Step4: check the condition i%2==0.

IF condition is TRUE, GO TO STEP 5.

ELSE,GO TO step 6.

Step5:print n.

Step6: stop the program.




```c
#include <stdio.h>

int main()
{
    int i, n;


    printf("Print all even numbers till: ");
    scanf("%d", &n);

    printf("Even numbers from 1 to %d are:
        \n", n);

    for(i=1; i<=n; i++)
    {

        if(i%2 == 0)
        {
            printf("%d\n", i);
        }
    }

    return 0;
}
```

```
/tmp/tMdjn6zmiB.o
Print all even numbers till: 10
Even numbers from 1 to 10 are:
2
4
6
8
10
```

4. Generation of ODD number series 1, 3, 5, …..

ALGORITHM:

Step1: start the program.

Step2: Give input or limit say n.

Step3: Using for loop,i<=n

IF condition is TRUE, Go To step 4.

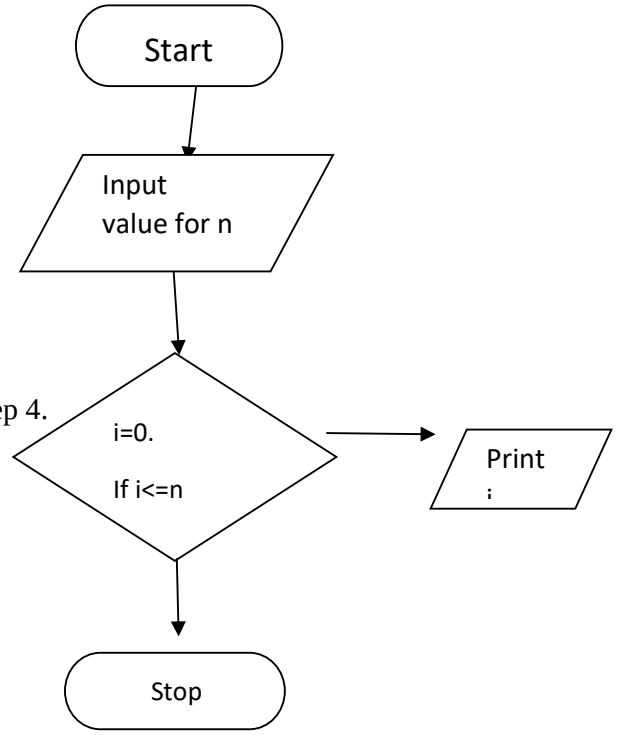
ELSE,Go To step 5.

Step4: if i%2 not equal to 0.

Then print i.

Else,go to step 5

Step5: stop the program



main.c

Run

Output

```c
#include <stdio.h>

int main()
{
    int i, n;


    printf("Print all Odd numbers till: ");
    scanf("%d", &n);

    printf("odd numbers from 1 to %d are: \n", n);

    for(i=1; i<=n; i++)
    {

        if(i%2 != 0)
        {
            printf("%d\n", i);
        }
    }
```

```
/tmp/PwBX6GGgXR.o
Print all Odd numbers till: 10
odd numbers from 1 to 10 are:
1
3
5
7
9
```

5 Generation of Fibonacci series 0, 1, 1, 2, 3, 5, 8, …..n

ALGORITHM:

Step1: start the program.

Step2: Give input say n.

Step3: input a=0,b=1,count=2

Step4: print a,b.

Step5:check count.

IF count>n,THEN GO TO step12.

Step6: next=a+b.

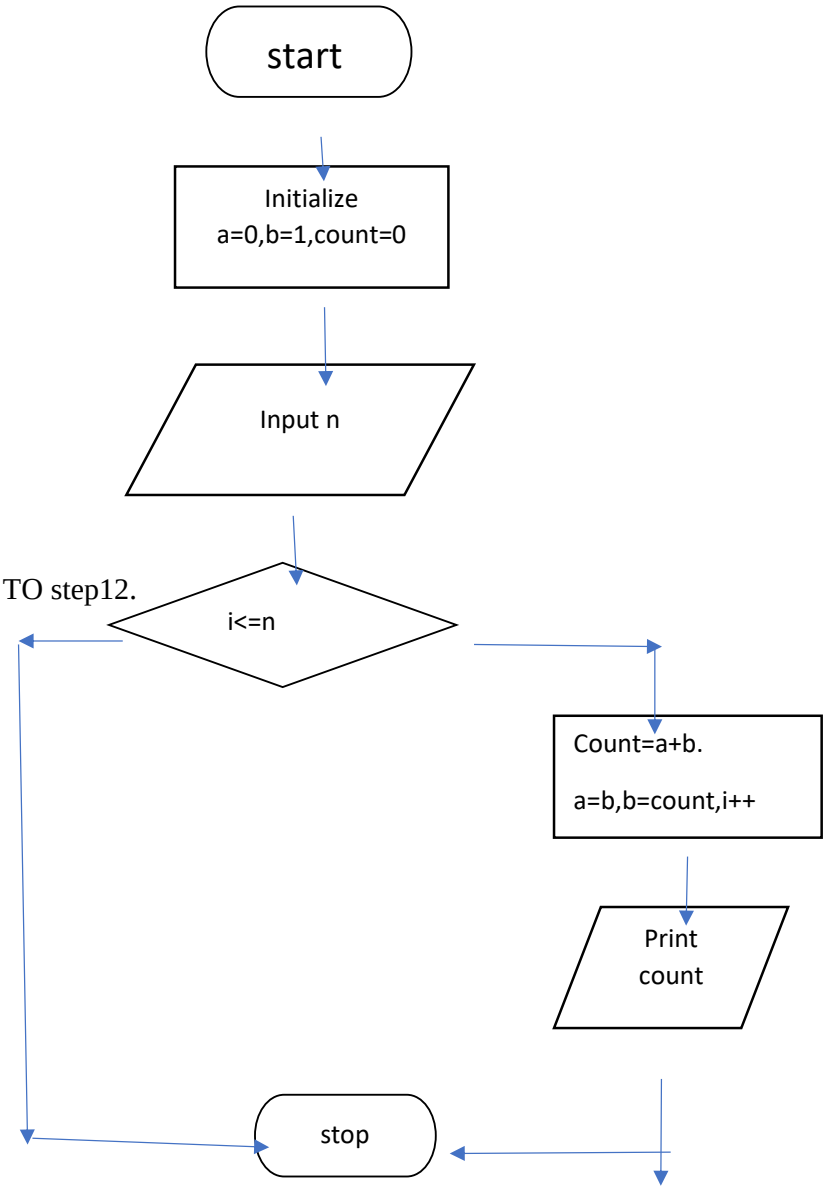
Step7:print next.

Step8:a=b.

Step9:b=next

Step10:count=count+1.

Step11:GO TO step 4.




```c
#include <stdio.h>
int main()
{
  int i, n;
  int t1 = 0, t2 = 1;
  int nextTerm = t1 + t2;
  printf("Enter the number of terms: ");
  scanf("%d", &n);
   printf("Fibonacci Series: %d, %d, ", t1,
       t2);
   for (i = 3; i <= n; ++i) {
    printf("%d, ", nextTerm);
    t1 = t2;
    t2 = nextTerm;
    nextTerm = t1 + t2;
  }
  return 0;
}
```

```
/tmp/tMdjn6zmiB.o
Enter the number of terms: 10
Fibonacci Series: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34,
```

6 Summing up Even Number series

ALGORITHM:

step1: start

step2: input value of N

step3: i=1,sum=0

step4: if(i>n)then go to step-8

ENDIF

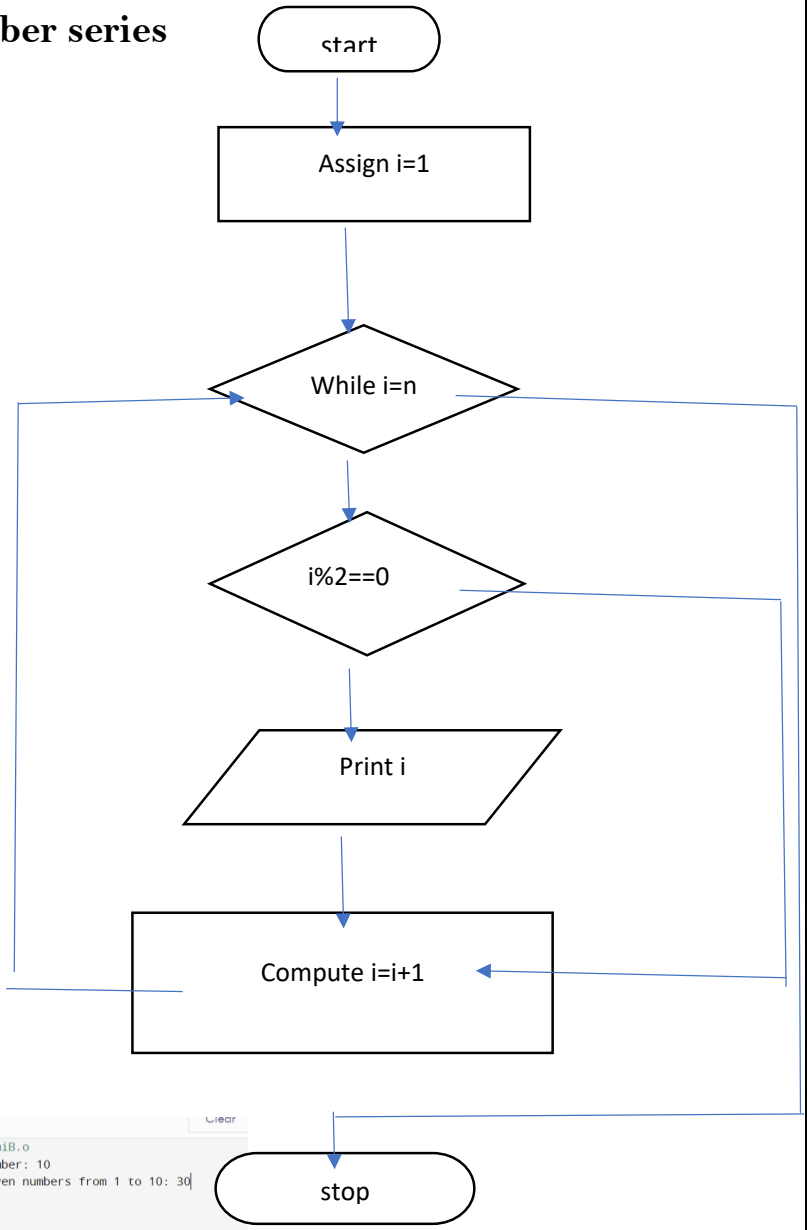
step5: Sum=sum+1

step6: i=i+1

step7: GO TO STEP-4

step8: display value of sum

step9:




```c
#include <stdio.h>
int main()
{
    int i, n, sum=0;
    printf("Enter any number: ");
    scanf("%d", &n);
    for(i=2; i<=n; i+=2)
    {
        sum += i;
    }
    printf("Sum of all even numbers from 1
        to %d: %d", n, sum);
    return 0;
}
```

```
/tmp/tMdjn6zmiB.o
Enter any number: 10
Sum of all even numbers from 1 to 10: 30
```

7 Summing up cubes of n numbers

ALGORITHM:

Step1: start the program.

Step2: Give input n .

Step3: initialize sum=0,m.

Step4: use for loop,i<=n.

If true go to step 5.

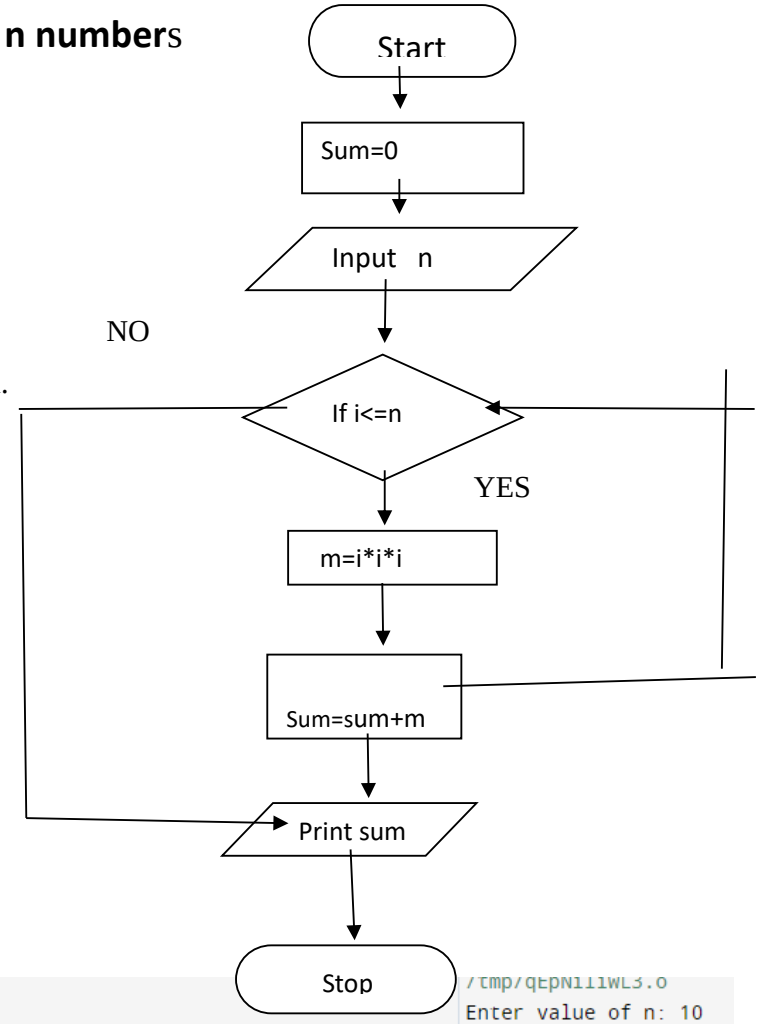
Else go to step7 .

step5:m=i*i*i.

Step6: sum=sum+m

Step8: print sum

Step9:stop




```c
#include<stdio.h>
long cube_sum_n_natural(int n) {
   long sum = 0;
   int i;
   for (i = 1; i <= n; i++) {
      sum += i * i * i; //cube i and add it with sum
   }
   return sum;
}
int main()
{
   int n;
   printf("Enter value of n: ");
   scanf("%d", &n);
   printf("Result is: %ld", cube_sum_n_natural(n));
return 0;
}
```

```
/tmp/qEpN111WL3.o
Enter value of n: 10
Result is: 3025
```

8 Finding whether the given integer is odd or even

ALGORITHM:

Step1: start

Step2: input say n.

Step3: using IF,check the condition.

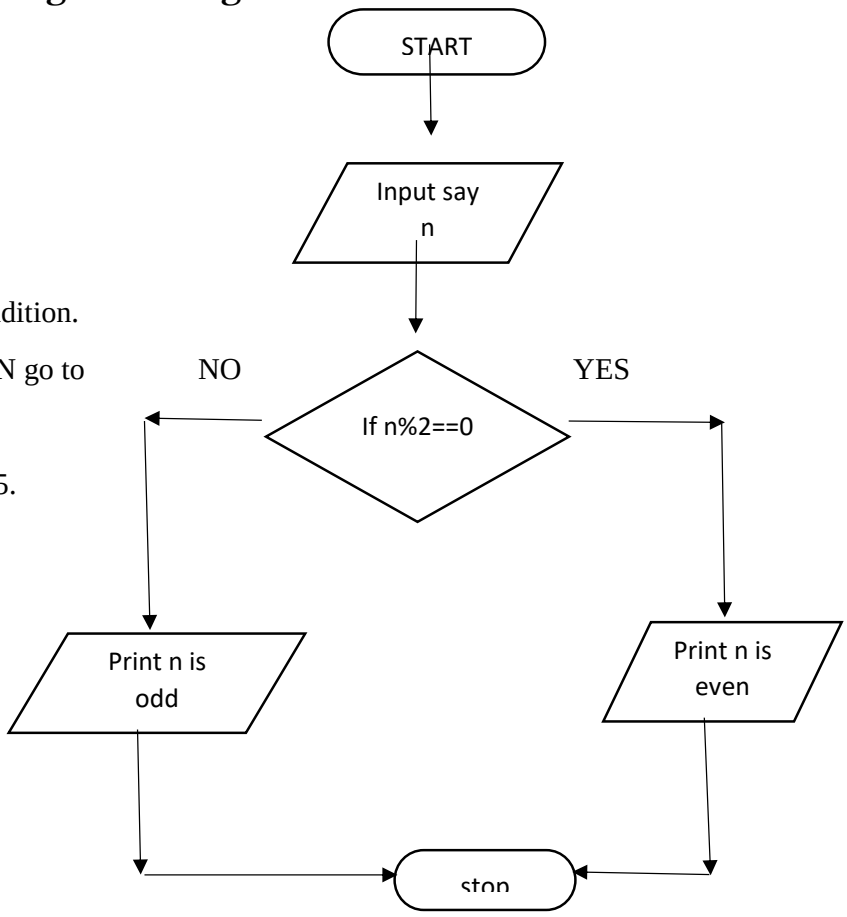
IF n%2==0,THEN go to step 4.

ELSE go to step 5.

Step4: print n is even.

Step5: print n is odd.

Step6: stop

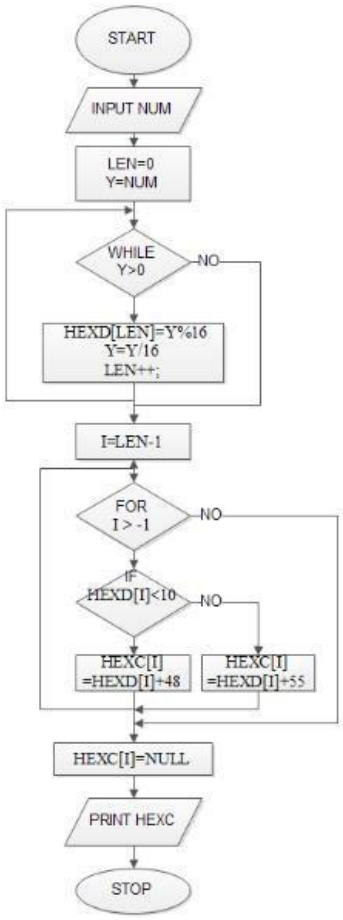



```c
#include <stdio.h>
int main() {
    int num;
    printf("Enter an integer: ");
    scanf("%d", &num);

    // true if num is perfectly divisible by 2
    if(num % 2 == 0)
        printf("%d is even.", num);
    else
        printf("%d is odd.", num);

    return 0;
}
```

```
/tmp/Sa4tb0Uhh4.o
Enter an integer: 2
2 is even.
```

DAY 2

1 program to convert decimal to hexa decimal



Algorithm:

Step 1: Input NUM

Step 2: LEN = 0 & Y=NUM

Step 3: While (Y > 0)

HEXD[LEN]=Y%16

Y=Y/16 LEN++

Step 4: for(I=LEN-1;I>-1;I–)

IF(HEXD[I]=HEXD[I]+48

ELSE

HEXC[I]=HEXD[I]+55;

Step 5: HEXC[I]=NULL

Step 6: Print HEX

```c
#include<stdio.h>
int main() {
long int num_decimal , remainder , quotient ;
int a = 1 , b , var ;
char hexanum_decimal[ 100 ] ;
printf( " Please enter decimal number here : " ) ;
scanf( "%ld" , &num_decimal ) ;
quotient = num_decimal ;
while( quotient != 0 ) {
var = quotient % 16 ;
if( var < 10 )
var = var + 48 ;
else
var = var + 55 ;
hexanum_decimal[ a++ ]= var ;
quotient = quotient / 16;
}
printf( " The equivalent hexadecimal value of decimal number is %ld : " ,
    num_decimal ) ;
for ( b = a -1 ; b > 0 ; b-- )
printf( "%c" , hexanum_decimal[ b ] ) ;
return 0 ;
}
```

```
/tmp/GZE3zr0onE.o
Please enter decimal number here : 23
The equivalent hexadecimal value of decimal number is 23 : 17
```

2)program to convert hexadecimal to decimal

Algorithm:

Step 1: start

Step2:dec=0,i=0

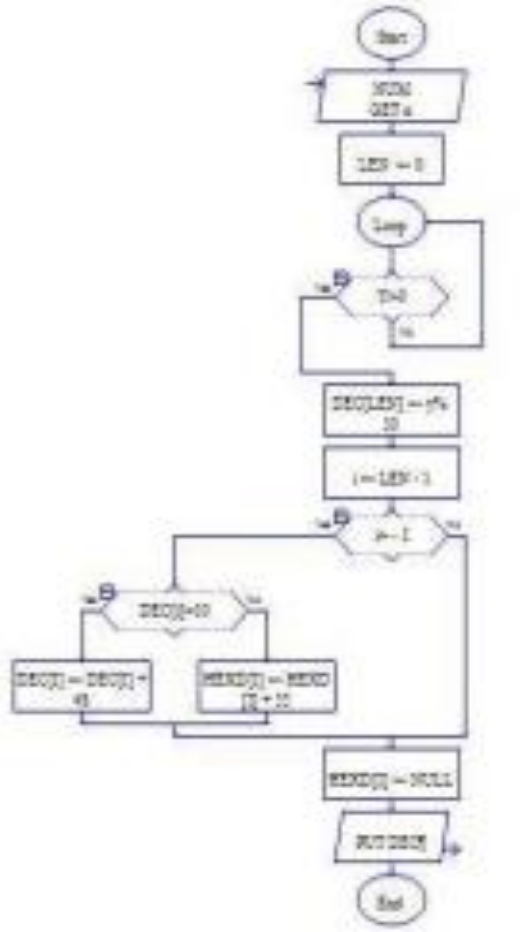
Step 3:enter the octal value

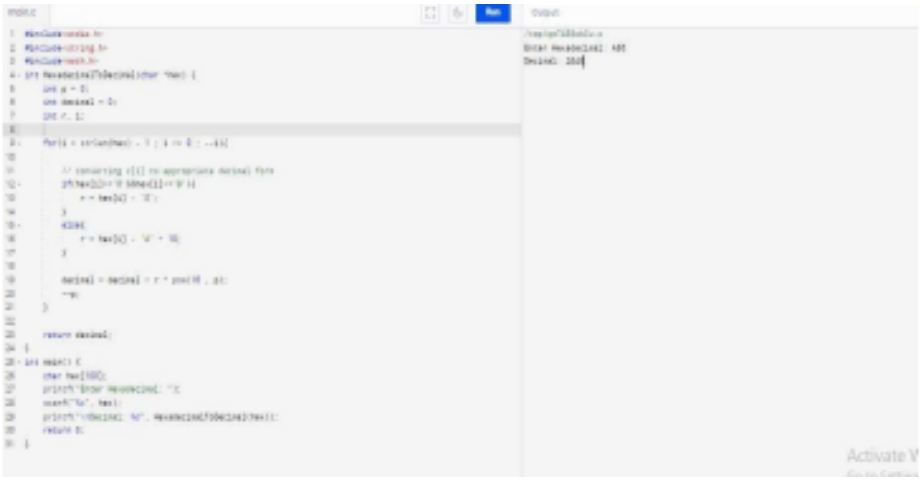
Step 4:if oct=0,then print dec

Step 5:if no dec=dec+(oct%i)

Step 6:print dec

Step 7:sto





:

3)program to convert decimal to octal

ALGORITHM:

Step 1:start

Step 2:read the decimal number from the user, say ‘d’

Step 3:initialise the octal number , octal =0

Step 4:initialise i=1

Step 5:repeat while d!=0:

Step 5.1: extract the remainder by: remainder =d%8

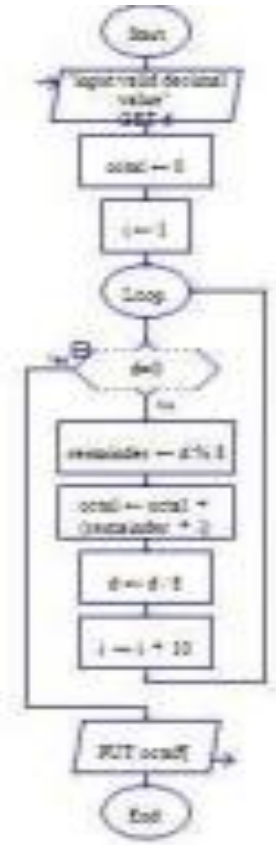
Step 5.2: octal=octal+(remainder*i)

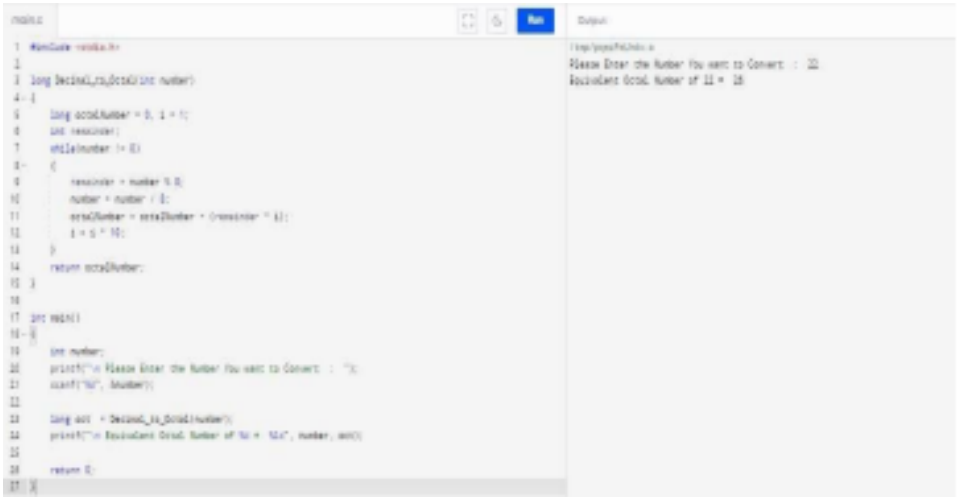
Step 5.3: d= d/8

Step 5.4: i=i*10

Step 6:display the octal number

Step 7:stop





4) Program to convert octal to decimal

ALOGRITHM

Step 1: Start

Step 2: Read the decimal number from the user, say ‘d’

Step 3: Initialise the octal number, octal=0

Step 4: Initialise i=1

Step 5: Repeat while d != 0:

Step 5.1: Extract the remainder by: remainder = d % 8

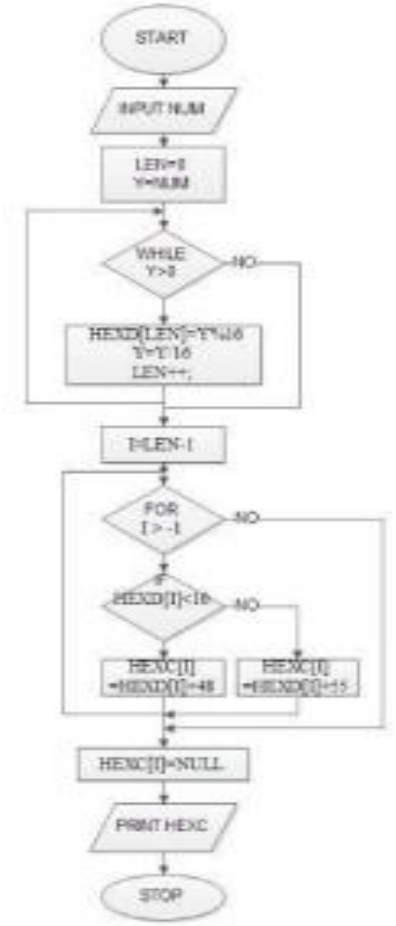
Step 5.2: octal = octal+ (remainder * i)

Step 5.3: d = d/8

Step 5.4: i = i * 10

Step 6: Display the octal number

Step 7: Stop



5) Program to convert binary to decimal

ALOGRITHM:

Step 1: Start

Step 2: Read the binary number from the user, say ‘n’

Step 3: Initialize the decimal number, d=0

Step 4: Initialize i=0

Step 5: Repeat while n != 0:

Step 5.1: Extract the last digit by: remainder = n % 10

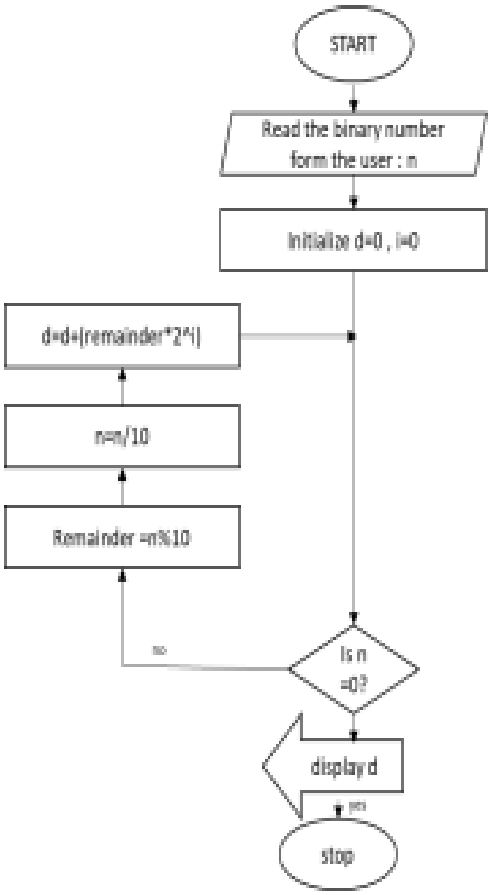
Step 5.2: n = n/10

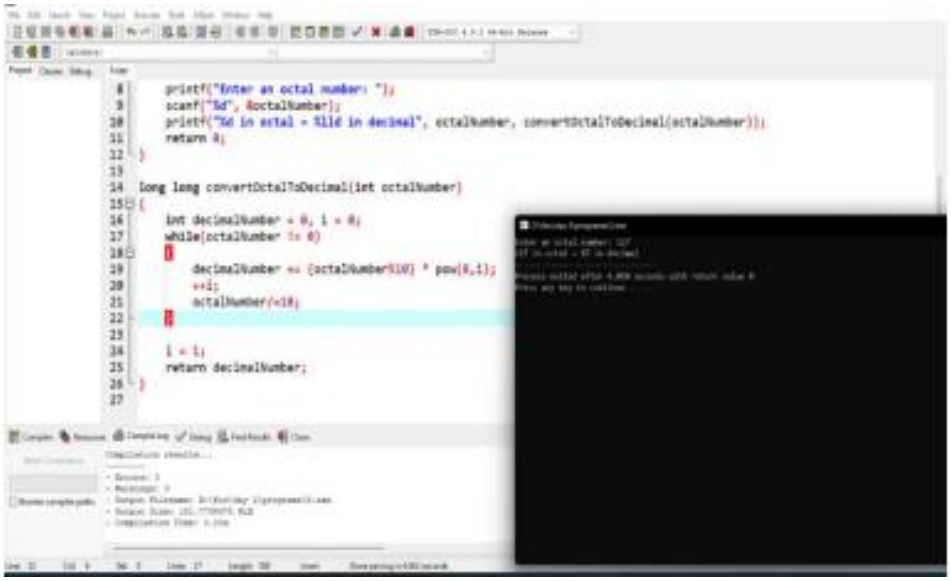
Step 5.3: d = d + (remainder * 2^i)

Step 5.4: Increment i by 1

Step 6: Display the decimal number, d

Step 7: Stop





6)write a program for binary addition

ALGORITHM:

Step 1: start

Step 2: enter 1 st binary value n1

Step 3: enter 2 ndbinary value n2

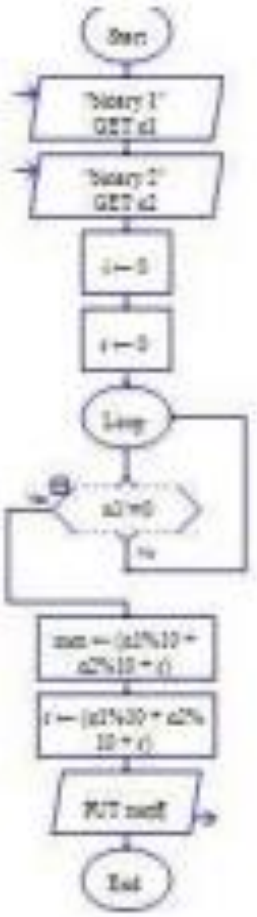
Step 4 : i=0 ,remainder r =0

Stepp :sum=(n1%10+n2%10+r) and r=(n1%10+n2%10+r)

Step 6: n1/10 and n2=n2/10

Step 7 :print sum

Step 8 :stop




```
#include <stdio.h>

int main()
{

    long binary1, binary2;
    int i = 0, remainder = 0, sum[20];

    printf("Enter the first binary number: ");
    scanf("%ld", &binary1);
    printf("Enter the second binary number: ");
    scanf("%ld", &binary2);
    while (binary1 != 0 || binary2 != 0)
    {
        sum[i++] = (binary1 % 10 + binary2 % 10 + remainder) % 2;
        remainder = (binary1 % 10 + binary2 % 10 + remainder) / 2;
        binary1 = binary1 / 10;
        binary2 = binary2 / 10;
    }
    if (remainder != 0)
        sum[i++] = remainder;
    --i;
    printf("Sum of two binary numbers: ");
    while (i != 0)
        printf("%d", sum[i--]);
    return 0;
}
```

```
Enter the first binary number: 10101
Enter the second binary number: 10101
Sum of two binary numbers: 101010
```

7)write a program for binary subtraction

ALGORITHM:

Step 1: start

Step 2: get binary n1,n2

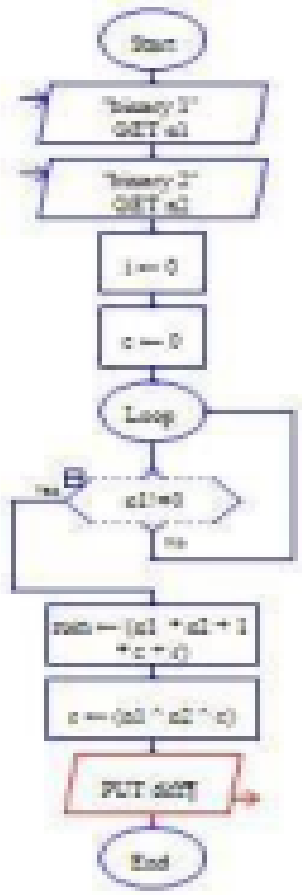
Step 3: i=0,c=0

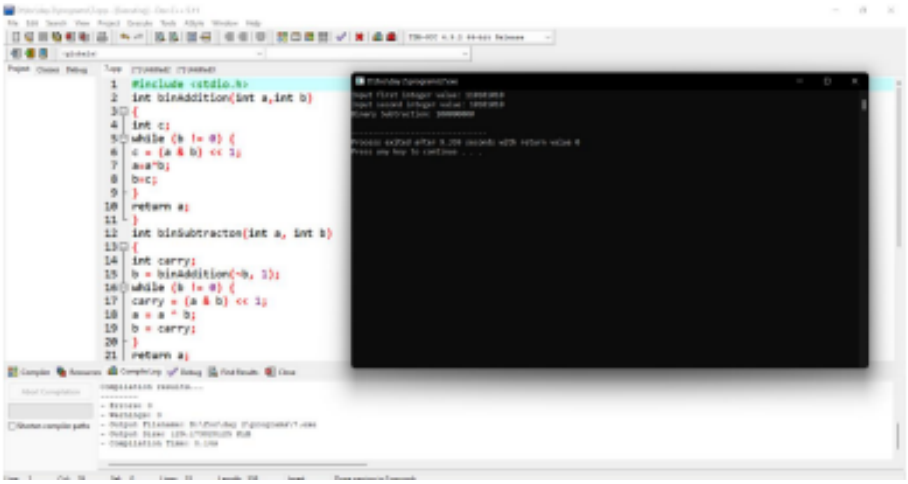
Step 4: n!=0

Step 5: diff=(n1^n2*c)

Step 6: print diff

Step 7: stop





8)write a program for binary multiplication

ALGORITHM:

Step 1: start

Step 2: input value of NUM

Step 3: i=1

Step 4: if (i>10)then

Go to step 9

End if

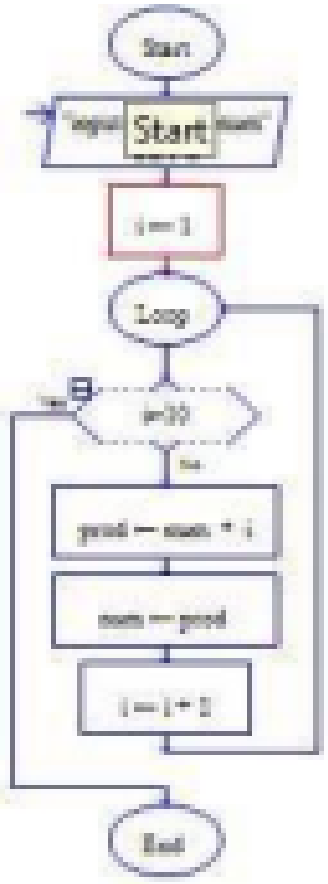
Step 5: prod=num*i

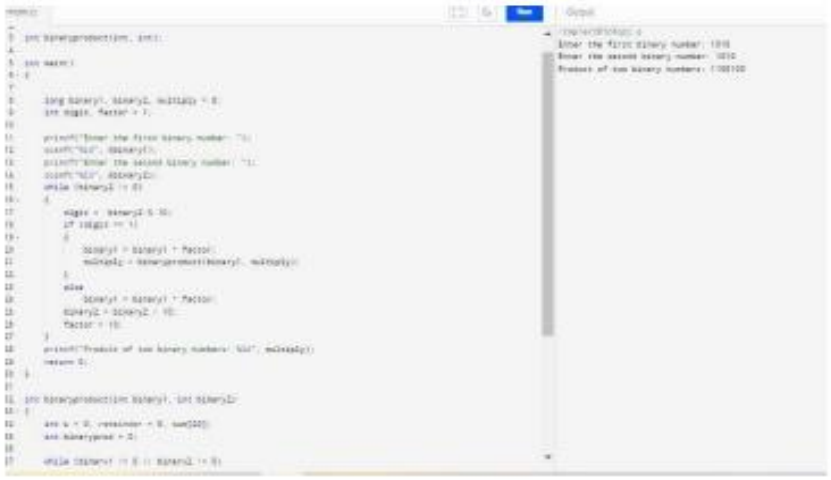
Step 6: write I “x”num “=” prod

Step 7: i=i+1

Step 8: go to step 4

Step 9: stop





DAY 3

Linux commands and shell programming

Linux commands:

1) Linux Directories command

1. pwd Command

The pwd command is used to display the location of the current working directory.

Syntax:

Pwd

2. mkdir Command

The mkdir command is used to create a new directory under any directory.

Syntax:

mkdir **<directory** name

3. rmdir Command

The rmdir command is used to delete a directory.

Syntax:

rmdir **<directory** name**>**

4. ls Command

The ls command is used to display a list of content of a directory.

Syntax:

ls

5. cd Command

The cd command is used to change the current directory.

Syntax:

cd **<directory** name**>**

2) Linux file commands

6. touch Command

The touch command is used to create empty files. We can create multiple empty files by executing it once.

Syntax:

1. touch **<file** name**>**
2. touch **<file1> <file2>** ....

7. cat Command

The cat command is a multi-purpose utility in the Linux system. It can be used to create a file, display content of the file, copy the content of one file to another file, and more.

Syntax:

1. cat [OPTION]... [FILE]..

8. rm Command

2. The rm command is used to remove a file.
3. **Syntax:**
4. rm <file name>

9. cp Command

The cp command is used to copy a file or directory.

Syntax:

To copy in the same directory:

1. cp **<existing** file name**> <new** file name**>**

10. mv Command

The mv command is used to move a file or a directory form one location to another location.

Syntax:

1. mv **<file** name**> <directory** path**>**

11. rename Command

The rename command is used to rename files. It is useful for renaming a large group of files.

Syntax:

1. rename 's/old-name/new-name/' files

3) Linux man commands

1. No Option: It displays the whole manual of the command.

Syntax :

```
$ man [COMMAND NAME]
```

2. Section-num: Since a manual is divided into multiple sections so this option is used to display only a specific section of a manual.

Syntax :

```
$ man [SECTION-NUM] [COMMAND NAME]
```

3. -f option: One may not be able to remember the sections in which a command is present. So this option gives the section in which the given command is present.

Syntax:

```
$ man -f [COMMAND NAME]
```

4. -a option: This option helps us to display all the available intro manual pages in succession.

Syntax:

```
$ man -a [COMMAND NAME]
```

5. -k option: This option searches the given command as a regular expression in all the manuals and it returns the manual pages with the section number in which it is found.

Syntax:

```
$ man -k [COMMAND NAME]
```

6. -w option: This option returns the location in which the manual page of a given command is present.

Syntax:

```
$ man -w [COMMAND NAME]
```

7. -I option: It considers the command as case sensitive.

Syntax:

```
$ man -I [COMMAND NAME]
```

Linux File Content Commands

12. head Command

The head command is used to display the content of a file. It displays the first 10 lines of a file.

Syntax:

1. head <**file** name>

13. tail Command

The tail command is similar to the head command. The difference between both commands is that it displays the last ten lines of the file content. It is useful for reading the error message.

9. wc Command

The wc command is used to count the lines, words, and characters in a file.

Syntax:

1. wc <**file** name>

10. od Command

The od command is used to display the content of a file in different s, such as hexadecimal, octal, and ASCII characters.

Syntax:

1. od -b <**fileName**> // Octal format
2. od -t x1 <**fileName**> // Hexa decimal format
3. od -c <**fileName**> // ASCII character format

11. sort Command

The sort command is used to sort files in alphabetical order.

Syntax:

1. sort <**file** name>

16. less Command

The less command is similar to the more command. It also includes some extra features such as 'adjustment in width and height of the terminal.' Comparatively, the more command cuts the output in the width of the terminal.

Syntax:

1. less <**file** name>

Linux Filter Commands

1. cat Command

The cat command is also used as a filter. To filter a file, it is used inside pipes.

Syntax:

1. cat <**fileName**> | cat or tac | cat or tac |...

2. cut Command

The cut command is used to select a specific column of a file. The '-d' option is used as a delimiter, and it can be a space (' '), a slash (/), a hyphen (-), or anything else. And, the '-f' option is used to specify a column number.

Syntax:

1. cut -d(delimiter) -f(columnNumber) <**fileName**>

3. grep Command

The grep is the most powerful and used filter in a Linux system. The 'grep' stands for "**global regular expression print**." It is useful for searching the content from a file. Generally, it is used with the pipe.

Syntax:

1. command | grep <searchWord>

4. comm Command

The 'comm' command is used to compare two files or streams. By default, it displays three columns, first displays non-matching items of the first file, second indicates the non-matching item of the second file, and the third column displays the matching items of both files.

Syntax:

1. comm <file1> <file2>

5. sed command

The sed command is also known as **stream editor**. It is used to edit files using a regular expression. It does not permanently edit files; instead, the edited content remains only on display. It does not affect the actual file.

Syntax:

1. command | sed 's/<oldWord>/<newWord>/'

6. sed command

The sed command is also known as **stream editor**. It is used to edit files using a regular expression. It does not permanently edit files; instead, the edited content remains only on display. It does not affect the actual file.

Syntax:

1. command | sed 's/<oldWord>/<newWord>/'

7. tr Command

The tr command is used to translate the file content like from lower case to upper case.

Syntax:

1. command | tr <'old'> <'new'>

8. uniq Command

The uniq command is used to form a sorted list in which every word will occur only once.

Syntax:

1. command <fileName> | uniq

Syntax:

1. host <domain name> or <ip address>

5) Linux file hierarchy commands

1. **/ (Root):** Primary hierarchy root and root directory of the entire file system hierarchy.
   - Every single file and directory starts from the root directory
   - The only root user has the right to write under this directory
   - /root is the root user's home directory, which is not the same as /
2. **/bin :** Essential command binaries that need to be available in single-user mode; for all users, e.g., cat, ls, cp.
   - Contains binary executables
   - Common linux commands you need to use in single-user modes are located under this directory.
   - Commands used by all the users of the system are located here e.g. ps, ls, ping, grep, cp
3. **/boot :** Boot loader files, e.g., kernels, initrd
   - Kernel initrd, vmlinux, grub files are located under

Linux I/O command

Overwrite

Commands with a single bracket **'>' overwrite** existing file content.

- **>** : standard output
- < : standard input
- 2> : standard error

Note: Writing **'1>'** or **'>'** and **'0<'** or **'<'** is same thing. But for stderr you have to write **'2>'**.

Syntax:

1. cat > <fileName>

Append

Commands with a double bracket **'>>' do not overwrite** the existing file content.

- >> - standard output
- << - standard input
- 2>> - standard error

Syntax:

1. cat >> <fileName>

Linux regex command

Linux shell commands **type command**

Linux 'type' command tell us whether a command given to the shell is a built-in or external command.

Syntax:

1. type **<command>**

type -a

The 'type -a' option tells about all type of command whether it is built-in, external, or aliased. Some commands are both external and built-in commands. But built-in command will always takes priority until and unless path of external command is mentioned.

Syntax:

1. type -a **<command>** **which**

Linux 'which' command locates the path of a command.

Syntax:

1. which **<command1>** **<command2>** **<command3>**...

12) Linux vi editor commands

8) Linux unix Tool commands

1. pwd command

Use the **pwd** command to find out the path of the current working directory (folder) you're in. The command will return an absolute (full) path, which is basically a path of all the directories that starts with a forward slash **(/)**. An example of an absolute path is **/home/username**.

2. cd command

To navigate through the Linux files and directories, use the **cd** command. It requires either the full path or the name of the directory, depending on the current working directory that you're in.
Let's say you're in **/home/username/Documents** and you want to go to **Photos**, a subdirectory of **Documents**. To do so, simply type the following command: **cd Photos**.
Another scenario is if you want to switch to a completely new directory, for example **/home/username/Movies**. In this case, you have to type **cd** followed by the directory's absolute path: **cd /home/username/Movies**.
There are some shortcuts to help you navigate quickly:

- **cd..** (with two dots) to move one directory up
- **cd** to go straight to the home folder

nslookup	Check domain name and IP information
shred	Delete a file by over writing its content
cat	Display, copy or combine text files
pwd>	Print path of current working directory
locate	Finding files by name on system
chown	Change ownership of a file
>alias	To short a command

- **cd**- (with a hyphen) to move to your previous directory

On a side note, Linux's shell is case sensitive. So, you have to type the name's directory exactly as it is.

3. ls command

The **ls** command is used to view the contents of a directory. By default, this command will display the contents of your current working directory. If you want to see the content of other directories, type **ls** and then the directory's path. For example, enter **ls /home/username/Documents** to view the content of **Documents**.

There are variations you can use with the **ls** command:

- **ls -R** will list all the files in the sub-directories as well
- **ls -a** will show the hidden files
- **ls -al** will list the files and directories with detailed information like the permissions, size, owner, etc.

4. cat command

cat (short for concatenate) is one of the most frequently used commands in Linux. It is used to list the contents of a file on the standard output (sdout). To run this

Shell programming

14)Shell Program to read a number and find its square

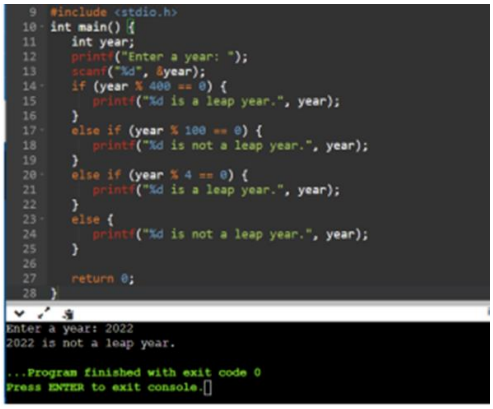
```
echo -n "Enter the value of a:"
read a
square=$(($a*$a))
echo "square of the value a=$square "
```

15)Shell Program to find the biggest of three numbers

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

if [ $num1 -gt $num2 ] && [ $num1 -gt $num3 ]
then
    echo $num1
elif [ $num2 -gt $num1 ] && [ $num2 -gt $num3 ]
then
    echo $num2
else
```

16)Shell Program to find leap year

```
#include <stdio.h>

 int main() {

 int year;

 printf("Enter a year: ");

 scanf("%d", &year);

if (year % 400 == 0) {

    printf("%d is a leap year.", year);

 }

 else if (year % 100 == 0) {

    printf("%d is not a leap year.", year);

 }

 else if (year % 4 == 0) {

    printf("%d is a leap year.", year);

}

 else {

    printf("%d is not a leap year.", year);

 }

 return 0;

 }
```



15)Shell Program to find the biggest of three numbers echo "---FIND THE GREATEST AMONG THREE NUMBERS---

" echo "Enter 1st number:" read first_num

echo "Enter 2nd number:"

read second_num

echo "Enter 3rd number:"

read third_num

if test $first_num -gt $second_num && test $first_num -gt $third_num

then

echo $first_num is the greatest number

. elif test $second_num -gt $third_num

Then

echo $second_num is the greaatest number.

Else

echo $third_num is the greatest number.

if

```
echo "---FIND THE GREATEST AMONG THREE NUMBERS---"
echo "Enter 1st number:"
read first_num
echo "Enter 2nd number:"
read second_num
echo "Enter 3rd number:"
read third_num
if test $first_num -gt $second_num && test $first_num -gt $third_num
then
    echo $first_num is the greatest number.
elif test $second_num -gt $third_num
then
    echo $second_num is the greaatest number.
else
    echo $third_num is the greatest number.
if
```

input

liation failed due to following error(s).

```
9 | echo "---FIND THE GREATEST AMONG THREE NUMBERS---"
```

17)Shell Program to prepare mark list using elif statement

```
clear
echo ---------------------------------
echo '\tStudent Mark List'
echo ---------------------------------
echo Enter the Student name
read name
echo Enter the Register number
read rno
echo Enter the Mark1
read m1
echo Enter the Mark2
read m2
echo Enter the Mark3
read m3
echo Enter the Mark4
read m4
echo Enter the Mark5
read m5
tot=$(expr $m1 + $m2 + $m3 + $m4 + $m5)
avg=$(expr $tot / 5)
```

```
echo ---------------------------------
echo '\tStudent Mark List'
echo ---------------------------------
echo "Student Name    : $name"
echo "Register Number : $rno"
echo "Mark1           : $m1"
echo "Mark2           : $m2"
echo "Mark3           : $m3"
echo "Mark4           : $m4"
echo "Mark5           : $m5"
echo "Total          : $tot"
echo "Average         : $avg"
if [ $m1 -ge 35 ] && [ $m2 -ge 35 ] && [ $m3 -ge 35 ] && [
$m4 -ge 35 ] && [ $m5 -ge 35 ]
then
 echo "Result          : Pass"

 if [ $avg -ge 90 ]
 then
  echo "Grade           : S"
 elif [ $avg -ge 80 ]
 then
```

```
        echo "Grade          : A"
elif [ $avg -ge 70 ]
then
        echo "Grade          : B"
elif [ $avg -ge 60 ]
then
        echo "Grade          : C"
elif [ $avg -ge 50 ]
then
        echo "Grade          : D"
elif [ $avg -ge 35 ]
then
        echo "Grade          : E"
fi
else
 echo "Result         : Fail"
fi
echo ---------------------------------
```

```
clear
echo ------------------------------------
echo '\tStudent Mark List'
echo ------------------------------------
echo Enter the Student name
read name
echo Enter the Register number
read rno
echo Enter the Mark1
read m1
echo Enter the Mark2
read m2
echo Enter the Mark3
read m3
echo Enter the Mark4
read m4
echo Enter the Mark5
read m5
tot=$(expr $m1 + $m2 + $m3 + $m4 + $m5)
```

Input

ilation failed due to following error(s).

```
n.c:9:1: error: unknown type name 'clear'
 9 | clear
   | ^~~~~
n.c:10:6: error: expected '=', ',', ';', 'asm' or '__attribute__' before '--' token
10 | echo ------------------------------------
   |      ^~
```

18Shell Program to perform arithmetic operation on two number

```
read a b
echo "What do you want to do? (1 to 5)"
echo "1) Sum"
echo "2) Difference"
echo "3) Product"
echo "4) Quotient"
echo "5) Remainder"
echo "Enter your Choice"
echo "Enter Two Numbers"
read n
case "$n" in
1) echo "The Sum of $a and $b is `expr $a + $b`";;
2) echo "The Difference between $a and $b is `expr $a -
$b`";;
3) echo "The Product of the $a and $b is `expr $a \*
$b`";;
4) echo "The Quotient of $a by $b is `expr $a / $b`";;
5) echo "The Remainder of $a by $b is `expr $a %
$b`";;
esac
```

```
 9  echo "Enter Two Numbers"
10  read a b
11  echo "What do you want to do? (1 to 5)"
12  echo "1) Sum"
13  echo "2) Difference"
14  echo "3) Product"
15  echo "4) Quotient"
16  echo "5) Remainder"
17  echo "Enter your Choice"
18  read n
19  case "$n" in
20  1) echo "The Sum of $a and $b is `expr $a + $b`";;
21  2) echo "The Difference between $a and $b is `expr $a - $b`";;
22  3) echo "The Product of the $a and $b is `expr $a \* $b`";;
23  4) echo "The Quotient of $a by $b is `expr $a / $b`";;
24  5) echo "The Remainder of $a by $b is `expr $a % $b`";;
25  esac
```

Input

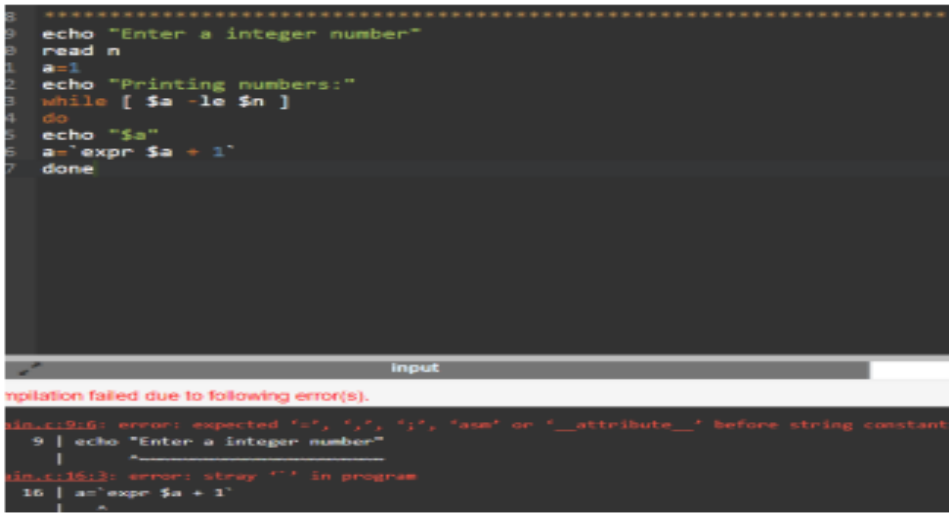
ompilation failed due to following error(s).

```
main.c:21:1: error: expected identifier or '(' before numeric constant
   21 | 2) echo "The Difference between $a and $b is `expr $a - $b`";;
      | ^
```

DAY 4

1.Shell Program to print n natural number

```
echo "Enter a integer number"
read n
a=1
echo "Printing numbers:"
while [ $a -le $n ]
do
echo "$a"
a=`expr $a + 1`
done
```



2.Shell Program to find area of different shapes

```
echo "Enter the radius : "
read r
echo "Area of the Circle is"
echo "3.14 * $r * $r" | bc
```

```
8  ********************************************************************************
9  echo "Enter a integer number"
0  read n
1  a=1
2  echo "Printing numbers:"
3  while [ $a -le $n ]
4  do
5  echo "$a"
6  a=`expr $a + 1`
7  done
```

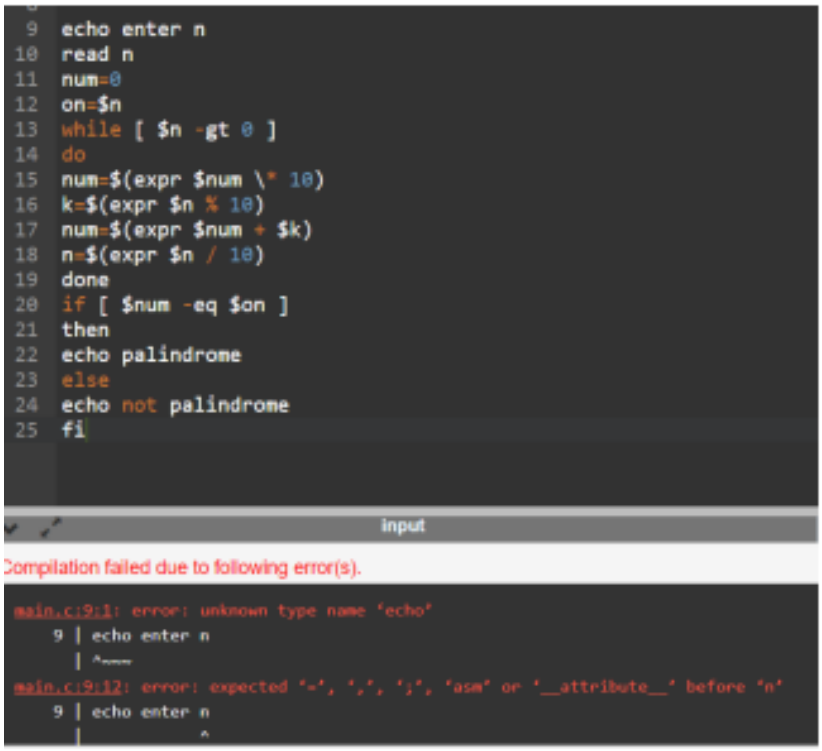
Input

npilation failed due to following error(s).

```
ain.c:9:6: error: expected '=', ',', ';', 'asm' or '__attribute__' before string constant
   9 | echo "Enter a integer number"
     |      ^~~~~~~~~~~~~~~~~~~~~~~
ain.c:16:3: error: stray '`' in program
  16 | a=`expr $a + 1`
```

3.Shell Program to check number is palindrome

```
echo enter n
read n
num=0
on=$n
while [ $n -gt 0 ]
do
num=$(expr $num \* 10)
k=$(expr $n % 10)
num=$(expr $num + $k)
n=$(expr $n / 10)
done
if [ $num -eq $on ]
then
echo palindrome
else
echo not palindrome
fi
```



4 Shell Program to solve quadratic equation using System;

```
class GFG
{

    // Method to check for solutions of equations
    static void checkSolution(int a, int b, int c)
    {

        // If the expression is greater
        // than 0, then 2 solutions
        if (((b * b) - (4 * a * c)) > 0)
            Console.WriteLine("2 solutions");

        // If the expression is equal to 0,
        // then 2 solutions
        else if (((b * b) - (4 * a * c)) == 0)
            Console.WriteLine("1 solution");

        // Else no solutions
        else
            Console.WriteLine("No solutions");
    }
        // Driver Code
        public static void Main()
        {
            int a = 2, b = 5, c = 2;
            checkSolution(a, b, c);
        }
        }
```

```
 9  echo enter n
10  read n
11  num=0
12  on=$n
13  while [ $n -gt 0 ]
14  do
15  num=$(expr $num \* 10)
16  k=$(expr $n % 10)
17  num=$(expr $num + $k)
18  n=$(expr $n / 10)
19  done
20  if [ $num -eq $on ]
21  then
22  echo palindrome
23  else
24  echo not palindrome
25  fi
```

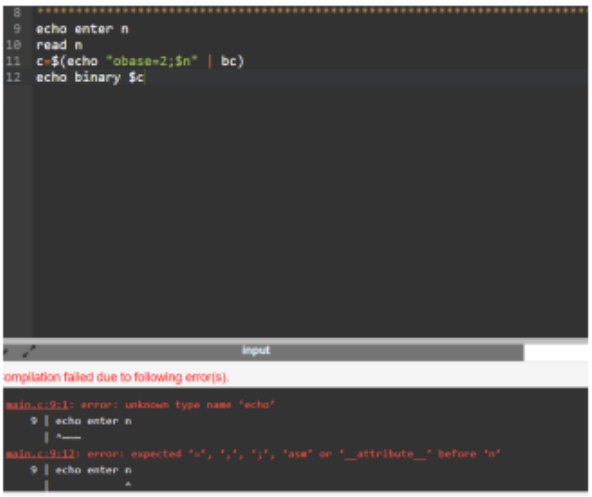
input

Compilation failed due to following error(s).

```
main.c:9:1: error: unknown type name 'echo'
    9 | echo enter n
      | ^~~~
main.c:9:12: error: expected '=', ',', ';', 'asm' or '__attribute__' before 'n'
    9 | echo enter n
      |            ^
```

5 Shell Program for decimal to binary conversion

```
echo enter n
read n
c=$(echo "obase=2;$n" |
echo binary $c
```



6 Shell Program factorial using recursion

```
#!/bin/sh

factorial()
{
if [ "$1" -gt "1" ]; then
 a=`expr $1 - 1`
 b=`factorial $a`
 c=`expr $1 \* $b`
 echo $c
else
 echo 1
fi
}

echo "Enter a number:"
 read x
 factorial $
```

```
 9  #!/bin/sh
10
11  factorial()
12  {
13  if [ "$1" -gt "1" ]; then
14   a=`expr $1 - 1`
15   b=`factorial $a`
16   c=`expr $1 \* $b`
17   echo $c
18  else
19   echo 1
20  fi
21  }
22
23  echo "Enter a number:"
24   read x
25   factorial $x
```

input

Compilation failed due to following error(s).

```
main.c:9:2: error: invalid preprocessing directive #!
    9 | #!/bin/sh
      |  ^
main.c:11:1: warning: return type defaults to 'int' [-Wimplicit-int]
   11 | factorial()
      | ^~~~~~~~~
```

7. Write a shell script to ask your name, program name an
enrollment number and print it on the screen

```
echo “Enter your name:”
read Name
echo “Enter your program name:”
read Prog
echo “Enter your
enrollmentnumber:”
read Enroll
clear
echo “Details you entered”
echo Name: $Name
echo Program Name: $Prog
echo Enrolment Number: $Enroll
```

```
7
8   **************************************************
9   echo"Enter your name:"
10  read name
11  echo"Enter your program name:"
12  read program
13  echo"Enter your enrollment number:"
14  read enroll
15  clear
16  echo"Details you entered"
17  echo Name: $Name
18  echo Program Name: $prog
19  echo Enrollment number: $Enroll
```

```
ile "main.py", line 1
 /**************************************************
 ^
taxError: invalid syntax
```

8. Write a shell script to find the sum, the average and the product of the four integers entered

```
echo Enter four integers with space between
read a b c d
Sum=`expr $a + $b + $c + $d`
Avg=`expr $sum / 4`21 2 3 4
Dec=`expr $sum % 4`
```

```
Dec=`expr \ ( $dec \* 1000 \ ) / 4`
Product=`expr $a \* $b \* $c \* $d`
echo Sum=$sum
echo Average=$avg.$dec
echo Product=$product
```

```
 9  echo Enter four integers with space between
10  read a b c d
11  Sum=`expr $a + $b + $c + $d`
12  Avg=`expr $sum / 4`21 2 3 4
13  Dec=`expr $sum % 4`
14  Dec=`expr \ ( $dec \* 1000 \ ) / 4`
15  Product=`expr $a \* $b \* $c \* $d`
16  echo Sum=$sum
17  echo Average=$avg.$dec
18  echo Product=$product
19
20
```

input

Compilation failed due to following error(s).

```
main.c:9:1: error: unknown type name 'echo'
    9 | echo Enter four integers with space between
      | ^~~~
main.c:9:12: error: expected '=', ',', ';', 'asm' or '__attribute__' before 'four'
    9 | echo Enter four integers with space between
```

9.Write a shell program to exchange the values of two variables

!/bin/bash

Program to swap two numbers

Static input of the

number

first=5

second=10

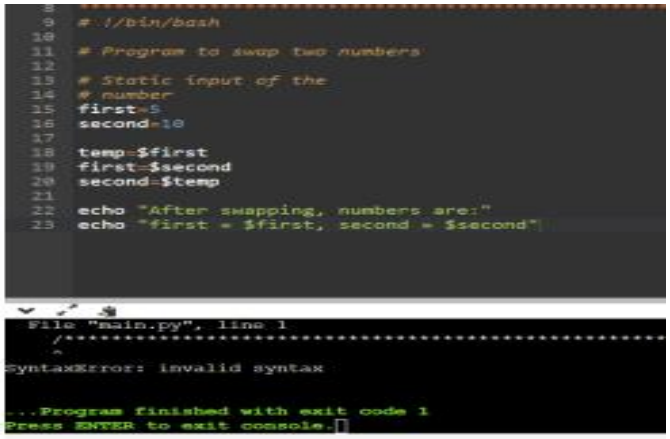
temp=$first

first=$second

second=$temp

echo "After swapping, numbers are:"

echo "first = $first, second = $second:”



10.) Write a shell script to display the digits which are in odd position in a given 5 digit number

echo "Enter a 5 digit number"

read num n=1

while [$n -le 5] do

a=`echo $num | cut -c $n`

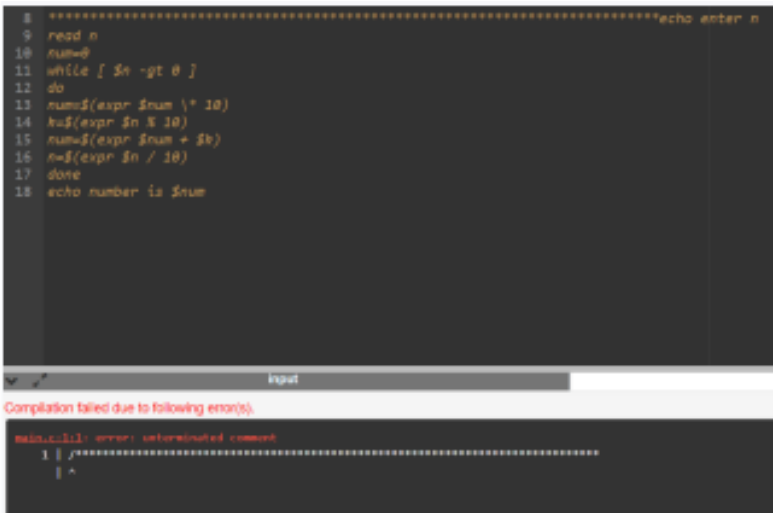
echo $a

n=`expr $n + 2`

done

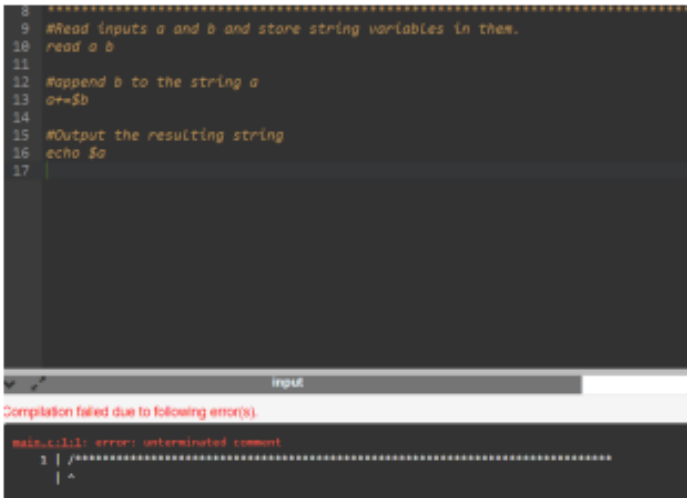
11. Write a shell program to reverse the digits of five digit integer

```
echo enter n
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 $sum
```



12.) Write a shell program to concatenate two strings and find the length of the resultant string

```
#Read inputs a and b and store strin
g variables in them.
read a b
#append b to the string a
a+=$b
#Output the resulting string
echo $a
```



13.Write a shell program to find the position of substring in given strin

g

script to get the substring position in given string

let give a string str="geeks for geeks is the best platform for computer science geeks"

now ask user to give the new string or use the given default string echo "Hello there, do you wanna give new string or use default

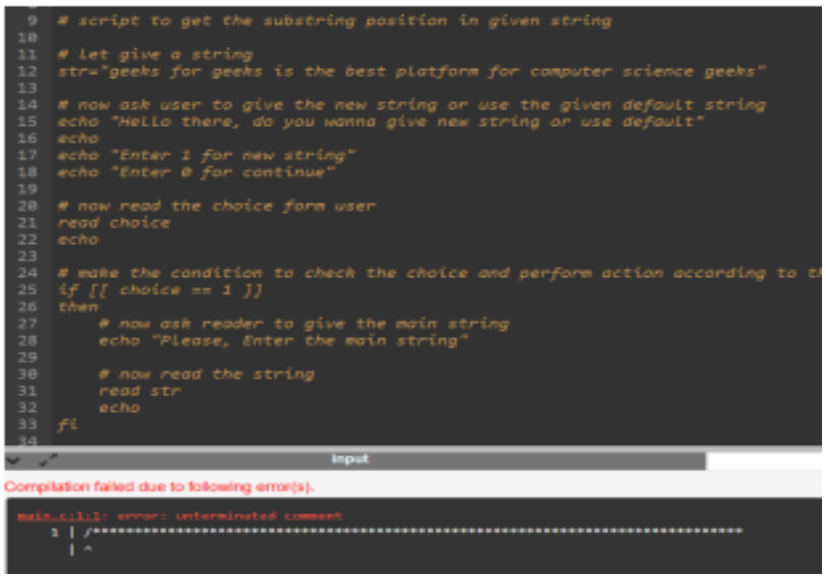
```
echo
echo "Enter 1 for new string"
echo "Enter 0 for continue"

# now read the choice form user
read choice
echo

# make the condition to check the choice and
perform action according to that
if [[ choice == 1 ]]
then
    # now ask reader to give the main string
    echo "Please, Enter the main string"

    # now read the string
    read str
    echo
fi


# print a message
echo "Let's continue to get the index of the
substring....."
echo
```

```
# make a loop to get the substring values
from the user
while [[ 1 ]]
do
    # print the statement
    echo "Enter a substring to get the
position of that string OR Enter -1 to get
exit"

    # now read the substr
    read substr

    # make a condition to check the value of
substr
    if [[ $substr != -1 ]]
    then
        # # 1st approach code to get the
substring position from given string ( 1st
approach )
        # # This approach is comparison on
char by char
        #
**********************************************
**************************

        # length of the given string
```

```
        lenGS=${#str}
        #length of the substr
        lenSS=${#substr}

        # check the condition where string
length is less than substring length
        if [[ $lenGS -lt $lenSS ]]
      then
            echo "Sorry, Your substring
exceed main string, Please Enter another"
            continue
        fi

        # variable to store position
        pos=-1
        # variable to check
        found=0

        # run three native loop ( brute force
approach )
        for (( i=0;i<lenGS;++i ))
        do

            if [[ ${str:i:1} == ${substr:0:1}
]]
```

```
            then

                # now loop to check that here
substring or not
                kstr=$i
                ksubstr=$i

                while (( kstr<lenGS &&
ksubstr<lenSS ))
                do
              if [[ ${str:kstr:1} !=
${substr:ksubstr:1} ]]
                    then
                        break
                    fi
                    kstr=`expr $kstr + 1`
                    ksubstr=`expr $ksubstr +
1`
                done

                # check if substring found
                if [[ ${ksubstr} == ${lenSS}
]]
                then
                    echo "Your substring
$substr is found at the index ${i+1}"
```

```
                    found=1
                    break
                fi
            fi
        done

        # check the substring found or not
        if [[ $found == 0 ]]
        then
            echo "Sorry, Your substring
$substr is not found in main string"
        fi
        echo

#*****************************************
************************

    else
        echo "okay! Closed"
        break
    fi
done
```



14. Write a shell program to find the gcd for the 2 given numbers

```
// Script for finding gcd of two number
// echo is for printing the message echo Enter two numbers with space in between
// read for scanning
read a b
// Assigning the value of a to m
m = $a
// Condition checking if b greater than m
// If yes the replace the value of m assign a new value
if [ $b -lt $m ]
then
m = $b
fi
// In do while loop we are checking the gcd
while [ $m -ne 0 ]
do
x = `expr $a % $m`
y = `expr $b % $m
// If x and y both are 0 then we complete over
// process and we print the gcd
if [ $x -eq 0 -a $y -eq 0 ] then
```

// Printing the greatest gcd of two given number

echo gcd of $a and $b is $m

break

fi

m = `expr $m - 1`

done

```
// Script for finding gcd of two number
// echo is for printing the message
echo Enter two numbers with space in between

// read for scanning
read a b

// Assigning the value of a to m
m = $a

// Condition checking if b greater than m
// If yes the replace the value of m assign a new value
if [ $b -lt $m ]
then
m = $b
fi

// In do while loop we are checking the gcd
while [ $m -ne 0 ]
```

input

ompilation failed due to following error(s).

```
main.c:1:1: error: unterminated comment
    1 | /*****************************************************************************
      | ^
```

15. Write a shell program to add, subtract and multiply the 2 given numbers passed as command line argument

!/bin/bash

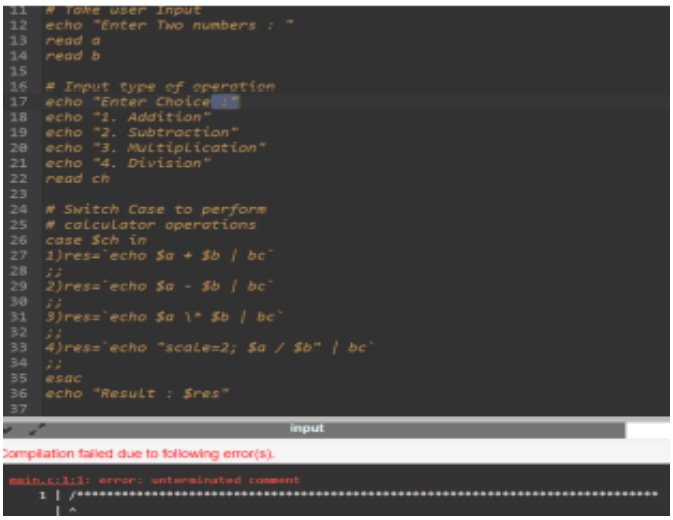
Take user Input echo "Enter Two numbers : "

read a

read b

Input type of operation

```
 echo "Enter Choice :"
echo "1. Addition"
echo "2. Subtraction"
echo "3. Multiplication"
echo "4. Division"
 read ch
 # Switch Case to perform
 # calculator operations
 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”
```



EVENTS PLAN

PROGRAM: SPORTS EVENT

EVENT VENUE AND TIME: SSE COLLEGE GROUND 9AM TO 5:30PM

STARTS	ENDS	AGENDA	C/NC
9:00AM	9:30AM	400 METERS RUNNING	☐
9:30AM	10:00AM	SHOTPUT	☐
10:00AM	10:30AM	HIGH JUMP	☐
10:30AM	11:00AM	1000M RUNNING	☐
11:00AM	11:30AM	JAVALIN THROW	☐
11:30AM	12:30PM	KABBADI	☐
1:30PM	2:30 PM	KHO KHO	☐
2:30PM	3:30PM	BADMANTION	☐
3:30PM	4:30PM	FOOTBALL	☐
4:30 PM	5:30PM	VOLLEY BALL	☐

SAVEETHA SCHOOL OF ENGINEERING

approved by AICTE |UK accreditation

Thandalam chennai - Tamil Nadu

SAVEETHA SCHOOL OF ENGINEERING
Approved by AICTE | IET-UK Accreditation

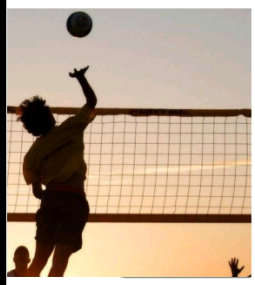
Department of computer science engineering
proudly presents
A state level sports
2022

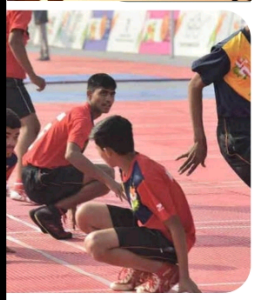
DETAILS

student co ordinators
- shiva - 9876543210
-charan - 7995609880
Staff co-ordinators
jay- 6303859390
jayanth - 9503276591

SPIORTS

Cricket
Kabaddi
coco
volly bll
Bat minton
Football





CULTURALS

REGISTRATION FEE:150/HEAD

FOR ALL EVENTS TIMINGS :9AMTO 5


INVITATION
THE PRINCIPAL,STAFF AND
STUDENTS OF COLLAGE
CORDIALLY INVITE YOU FOR
THE 15th ANNUAL COLLAGE DAY
on friday 10th june,2022 at 5:030pm
AVENUE: IN COLLAGE GROUND
Guest: VIJAY YADAV (AP CM)
Saveetha
school of
engineering
we shall be honured with
your distinguished presence
SAVEETHA
SCHOOL OF ENGINEERING
Approved by AICTE | IET-UK Accreditation
DR.NIMEL ROSS

CERTIFICATE

OF ACHIEVEMENT

SPORTS EVENT

This certificate is produdly present to :_____
Regonition of his/her outstanding participation ,hardwork and sportsmanship on specific sports team this year !

Signature and date

DR.NIMEL ROSS (principal)


Iphone 13 pro max
15% off
on sale