**Ruthul Jindal S**

**19bec1061**

**CSE assignment**

**1.**

Step 1:  Input N & M

Step 2:  While (N < M)

                         I=2

Step 4:                  While (I<N)

Step 5:                                 IF N%I == 0

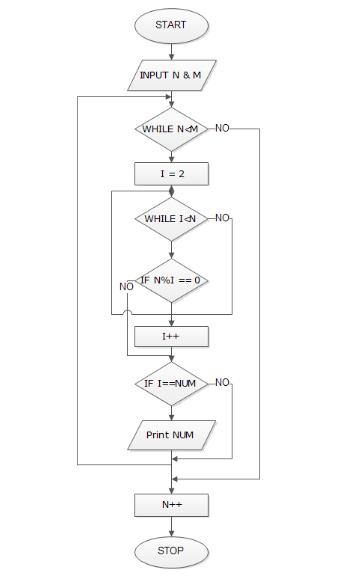
                                                     goto Step 7

Step 6:                           I++

Step 7:        IF I==NUM

                                Print NUM  
 Step 7:   N++

Flowchart:



**2.**

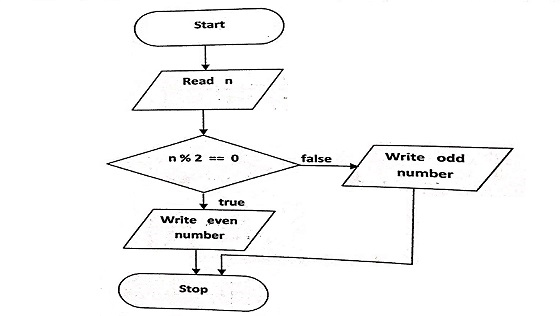
 Step 1 : Start

 Start 2 : Read n

 Start 3 : if (n%2 == 0), Write " even number" else Write " odd number"

 Start 4 : Stop

Flowchart:



**3.**

Step 1: Start

Step 2: Declare variables n,fact and i.

Step 3: Initialize variables factorial←1 i←1

Step 4: Read value of n

Step 5: Repeat the steps until i=n

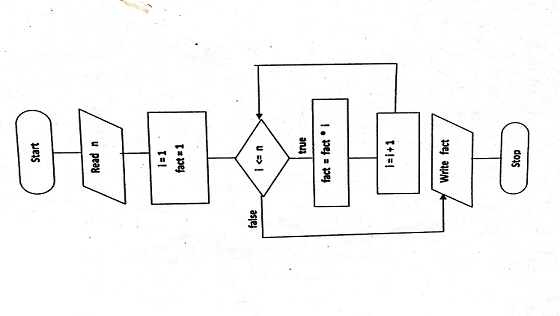
fact←fact\*i

i←i+1

Step 6: Display factorial

Step 7: Stop

Flowchart:



**4.**

Step 1: Start

Step 2: Declare variables a,b and show.

Step 3: Initialize variables a←0 b←1

Step 4: Display a and b

Step 5: Repeat the steps until b≤1000

show←b

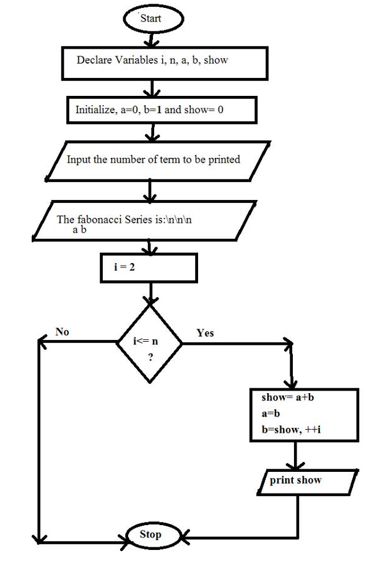
b←b+a

a←show

Display b

Step 6: Stop

Flowchart:



**5.**

Step 1: Start

Step 2: Declare Variable sum, temp, num

Step 3: Read num from User

Step 4: Initialize Variable sum=0 and temp=num

Step 5: Repeat Until num>=0

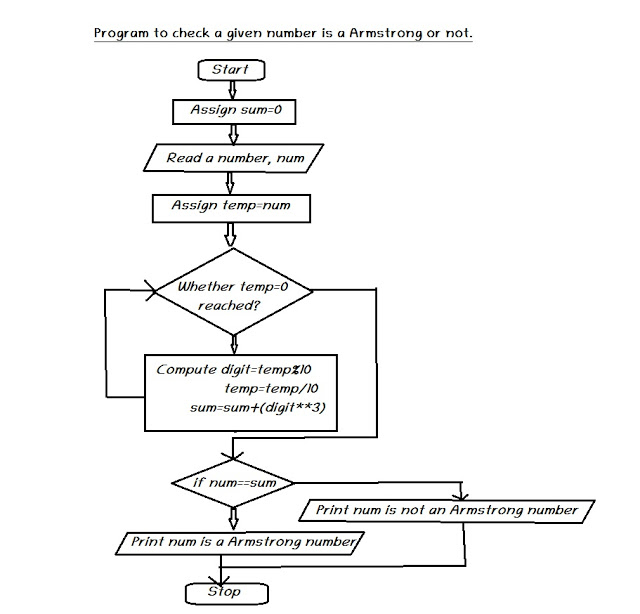
sum=sum + cube of last digit i.e [(num%10)\*(num%10)\*(num%10)]

num=num/10

Step 6: IF sum==temp Print "Armstrong Number" ELSE Print "Not Armstrong Number"

Step 7: Stop

Flowchart:



**6.**

Input: num

Step 1: Initialize rev\_num = 0

Step 2: Loop while num > 0

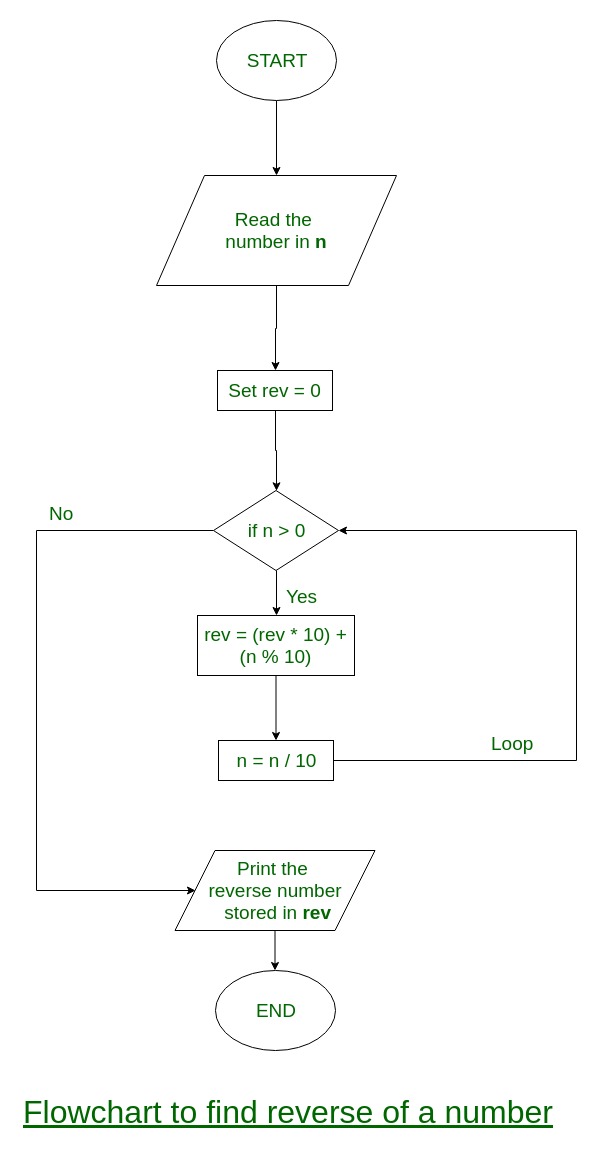
(a) Multiply rev\_num by 10 and add remainder of num divide by 10 to rev\_num rev\_num = rev\_num\*10 + num%10;

(b) Divide num by 10

Step 3: Return rev\_num

Step 4: End

Flowchart:



**7.**

Step 1: Start

Step 2: Declare matrix A[r][c];

and matrix B[r][c];

and matrix C[r][c]; rows= r columns= c

Step 3: Read r, c, p, A[][] and B[][]

Step 4: Declare variable i=0, j=0

Step 5: Repeat until i < r

5.1: Repeat until j < c

C[i][j]=A[i][j] + B[i][j]

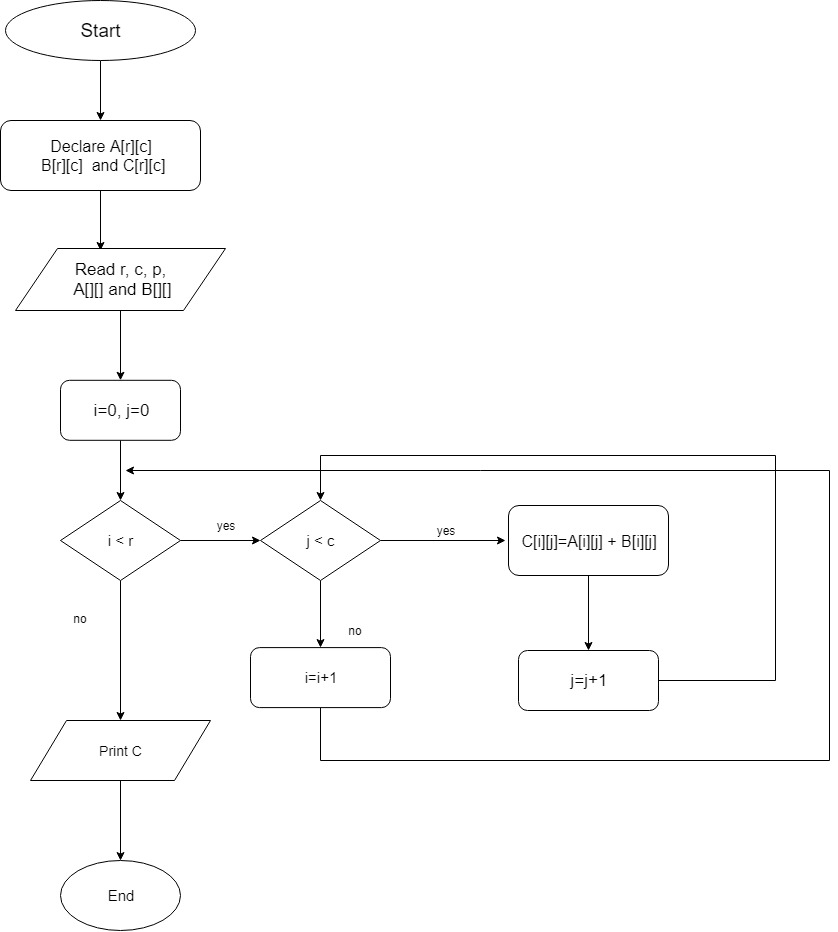
Set j=j+1

5.2: Set i=i+1

Step 6: C is the required matrix

Step 7: Stop

Flowchart:



**8.**

Step 1: integer a,b,c;

Step 2. float d,r1,r2,r3,r4,r;

Step 3. print ‘Enter values of a,b,c of a quadratic equation:’;

Step 4. read a,b,c;

Step 5. value of a,b,c is transferred to function ‘quad’and body of function is:

Step 6. d b x b-4 x a x c;

Step 7 .if(d<0)

Step 8 .print ‘Value of Discriminant is negative’;

Step 9. else

Step 10 .if(d=0)

Step 11. print ‘Roots are real’

Step 12. r=-b/2\*a;

Step 13. print ‘First and Second root of equation:’,r;

Step 14. else

Step 15. r1 -b+sqrt(d);

Step 16. r2 2\*a;

Step 17. r r1/r2;

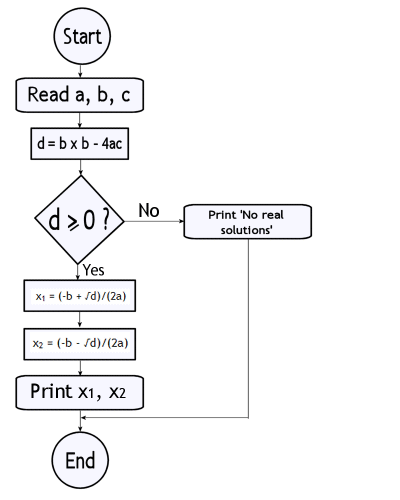
Step 18. r3 -b-sqrt(d);

Step 19. r4 r3/r2;

Step 20. print ‘First root of equation: ’, r;

Step 21. print ‘Second root of equation: ’, r4;

Flowchart :

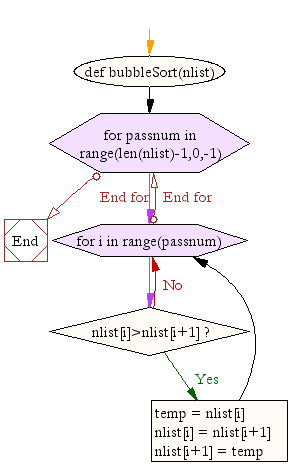


**9.**

Algorithm :

* Start
* Declare and define a function for sting comparison
* Enter the number of strings to be sorted
* Input all the strings
* Allocate the memory
* Call the function to sort the strings in alphabetical order
* Print the sorted strings
* Stop

Flowchart:



**10.**

**1**. **Start**;  
**2**. **Initialize the known variables:** max hours worked with no overtime (**MaxNoOvertime**), bonus rate for overtime hours (**BonusRate**),  
non taxable payroll amount (**MaxNoDue**), tax rate (**Due**);  
**3**. **Enter** hours worked overtime (**HoursWorked**) and hourly rate (**HourRate**);  
4. **If** (**HourWorked**-**MaxNoOvertime**)<=0 **Then**  
    **GrossPay=HoursWorked\*HourRate**;  
  **Else**  
    **GrossPay=HourRate\*(MaxNoOvertime + BonusRate\*(HoursWorked-MaxNoOvertime))**;  
5. **If** **GrossPay<=MaxNoDue Then**  
   **NetPay=GrossPay**;  
  **Else**  
    **NetPay=GrossPay-Due\*(GrossPay-MaxNoDue)**;  
6. **Display** **GrossPay**, **NetPay**;  
7. **End**.

**12.**

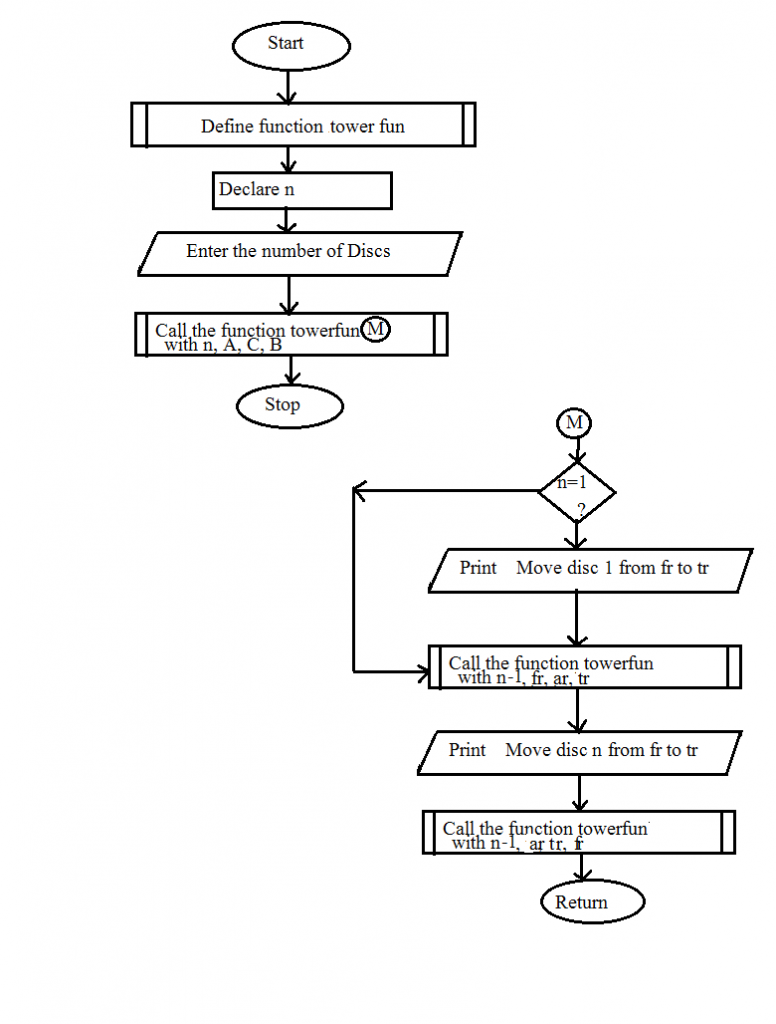
Algorithm:

* Start
* Define function
* Declare variable n
* Enter the number of disc
* Call the function
* Stop

**Function Definition: (Recursion)**

* if n=1
  1. print Move disc 1 from fr to tr
* Call the function with n-1, fr, ar, tr
* Move disc n from fr to tr
* Call function with n-1, ar, tr, fr
* Return

Flowchart:



**13.**

Algorithm:

Step 1: input height and weight of your body.

Step 2: BMI=(weight/height\*height)

Step 3: if BMI<16 print serious underweight

Step 4: if BMI is between 16 and 18 print underweight

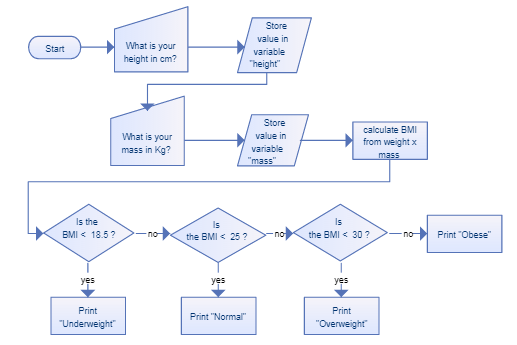
Step 5: if BMI is between 18 and 24 print normal weight

Step 6: if BMI is between 24 and 29 print overweight

Step 7: if BMI is between 29 and 35 print seriously overweight

Step 4: if BMI>35 print gravely overweight

Flowchart:



15.

Program:

L=[‘abc’,’xyz’,’aba’,’1221’]

G=len(L)

For I in L:

C=L[i]

If(C[0]==C[len©-1]):

J=J+1

Print J

16:

Program:

def last(n): return n[-1]

def sort\_list\_last(tuples):

return sorted(tuples, key=last)

print(sort\_list\_last([(2, 5), (1, 2), (4, 4), (2, 3), (2, 1)]))

17:

Program:

def pr ():

li = list()

for i in range(1,21):

li.append(i\*\*2)

print(li[:5])

print(li[-5:])

pr()

18:

Program:

board=['0','1','2','3','4','5','6','7','8']

empty = [0,1,2,3,4,5,6,7,8]

def display\_board():

    clear\_output()

    print('  |   |   ')

    print(board[0]+' | '+board[1]+' | '+board[2])

    print('  |   |   ')

    print('---------')

    print('  |   |   ')

    print(board[3]+' | '+board[4]+' | '+board[5])

    print('  |   |   ')

    print('---------')

    print('  |   |   ')

    print(board[6]+' | '+board[7]+' | '+board[8])

    print('  |   |   ')

def player\_input(player):

    player\_symbol = ['X','O']

    correct\_input = True

    #print('player {playerNo} chance! Choose field to fill {symbol}'.format(playerNo = player, symbol = player\_symbol[player]))

    position = int(input('player {playerNo} chance! Choose field to fill {symbol} '.format(playerNo = player +1, symbol = player\_symbol[player])))

    #print(type(position))

    """for eligible in empty:

        if position == eligible:

            correct\_input = True"""

    if board[position] == 'X' or board[position] == 'O':

        correct\_input = False

    if not correct\_input:

        print("Position already equipped")

        player\_input(player)

    else:

        empty.remove(position)

        board[position] = player\_symbol[player]

        return 1

def check\_win():

    player\_symbol = ['X','O']

    winning\_positions =[[0,1,2],[3,4,5],[6,7,8],[0,3,6],[1,4,7],[2,5,8],[0,4,8],[2,4,6]]

    for check in winning\_positions:

        first\_symbol = board[check[0]]

        if first\_symbol != ' ':

            won = True

            for point in check:

                if board[point] !=  first\_symbol:

                    won = False

                    break

            if won:

                if first\_symbol == player\_symbol[0]:

                    print('player 1 won')

                else:

                    print('player 2 won')

                break

        else:

            won = False

    if won:

        return 0

    else:

        return 1

def play():

    player = 0

    while empty and check\_win():

        display\_board()

        player\_input(player)

        player = int(not player)

    if not empty:

        print("NO WINNER!")

19:

Program :

import string

from random import \*

characters = string.ascii\_letters + string.punctuation + string.digits

password = "".join(choice(characters) for x in range(randint(8, 16)))

print password

20:

Program :

list1 = [10, 21, 4, 45, 66, 93, 11]

even\_nos = list(filter(lambda x: (x % 2 == 0), list1))

print("Even numbers in the list: ", even\_nos)

21:

Program :

Def check(n):

Lis=[1,12,15,47,59,68,88,99,101]

If(n in Lis):

Return True

Else:

Return false

22:

Program:

A=[1,1,3,5,8,13,21,34,55,89]

B=[1,2,3,4,5,6,7,8,9,10,11,12,13]

For i in range(0,len(A)-1):

For j in range(0,len(B)-1):

If(A[i]==B[j]):

Print(A[i]+” “)

23:

Program:

def countCurrency(amount):

notes = [2000, 500, 200, 100,

50, 20, 10, 5, 1]

noteCounter = [0, 0, 0, 0, 0,

0, 0, 0, 0]

print ("Currency Count -> ")

for i, j in zip(notes, noteCounter):

if amount >= i:

j = amount // i

amount = amount - j \* i

print (i ," : ", j)

24:

Program:

n=int(input("Enter number of years"))

total = n\*12\*8

f=total/4

m=total/3

son=total/2

print("father ",f)

print("mother ",m)

print("son ",son)

25:

Program:

import random

  guessesTaken = 0

  print('Hello! What is your name?')

  myName = input()

  number = random.randint(1, 20)

print('Well, ' + myName + ', I am thinking of a number between 1 and 20.').

 while guessesTaken < 63.     print('Take a guess.')

guess = input()

     guessint(guess)

     guessesTaken = guessesTaken + 1

     if guess < number:

         print('Your guess is too low.') # There are eight spaces in front of print.

    if guess > number:

         print('Your guess is too high.')

     if guess == number:

         break

 if guess == number:

     guessesTaken = str(guessesTaken)

     print('Good job, ' + myName + '! You guessed my number in ' + guessesTaken + ' guesses!')

 if guess != number:

     number = str(number)

print('Nope. The number I was thinking of was ' + number)

26:

Program:

N=int(input(‘Enter range’))

Odd=list()

Even = list()

For i in range(1,N):

If(i%2==0):

Even.append(i)

Else:

Odd.append(i)

Print(“odd “,Odd)

Print(“Even “,Even)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_