Ex. No 1 **COMPUTE THE GCD OF TWO NUMBERS.**

Date:

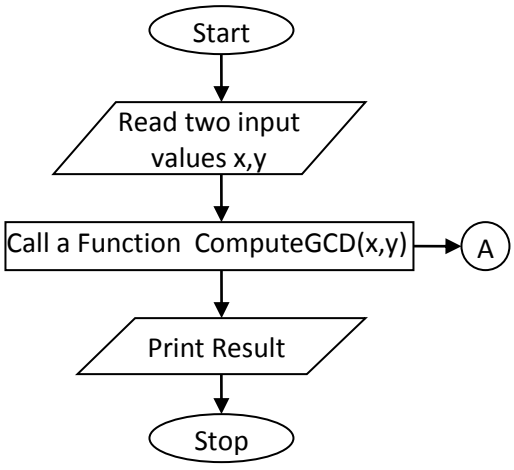
Aim:

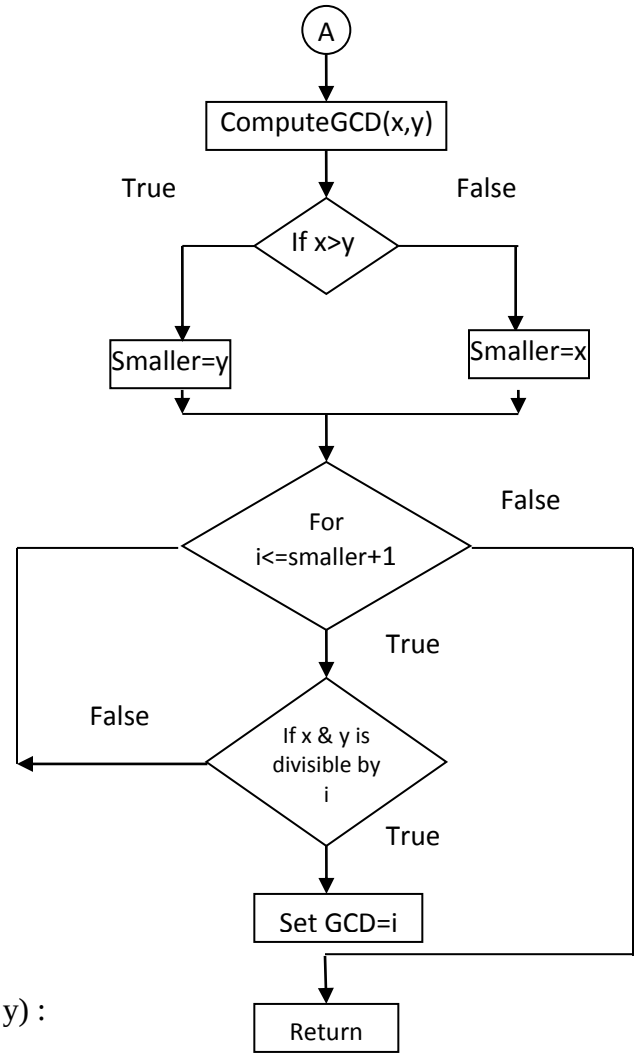
To compute the GCD of two numbers

Algorithm:

1. Read two input values using input function
2. Convert them into integers
3. Define a function to compute GCD
    a. Find smallest among two inputs
    b. Set the smallest
    c. Divide both inputs by the numbers from 1 to smallest+1
        If the remainders of both divisions are zero
            Assign that number to gcd
    d. Return the gcd
4. Call the function with two inputs
5. Display the result

Flow Chart:





Program :

```
def  computeGCD (x, y) :
        if  x < y :
                smaller = x
        else:
                smaller = y
        for i in range(1, smaller+1) :
                if   (x % i == 0)  and  (y % i == 0) :
                        gcd = i
        return  gcd

num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))
print("The G.C.D. of  ", num1,"  and  ", num2,"  is = ", computeGCD(num1, num2))
```

Output:

```
Enter first value: 12
Enter second value: 20
The GCD of 12 and 20 is = : 4
```

Result:

Thus, the program to compute the GCD of two numbers has been executed successfully.

Ex. No 2 FIND THE SQUARE ROOT OF A NUMBER (NEWTON'S METHOD)

Date:

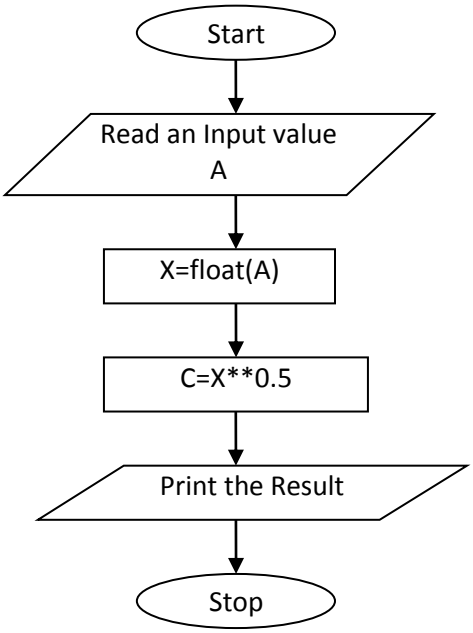
Aim:

To find the square root of a number (Newton's method)

Algorithm:

1. Read one input value using input function
2. Convert it into float
3. Find the square root of the given number using the formula inputvalue ** 0.5
4. Print the result
5. Exit.

Flow Chart:



Program :

```
A=input("enter a value")
x=float(A)
C=x**0.5
print("square root of   ", x , "is =",C)
```

Output:

```
C:\Python36-32\mypgm>ex2.py
Enter a value:  16
Square root of 16 is= 4.0
```

Result:

Thus, the program to To find the square root of a number (Newton's method) has been executed successfully.

Ex. No 3 **EXPONENTIATION (POWER OF A NUMBER)**

Date:

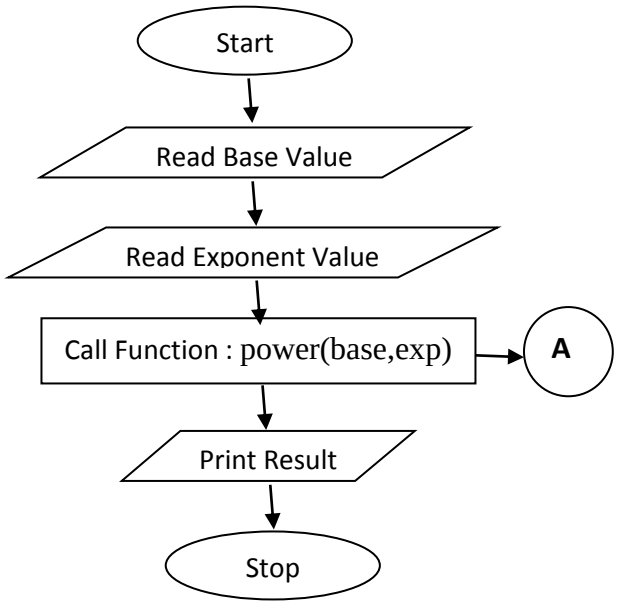
Aim:

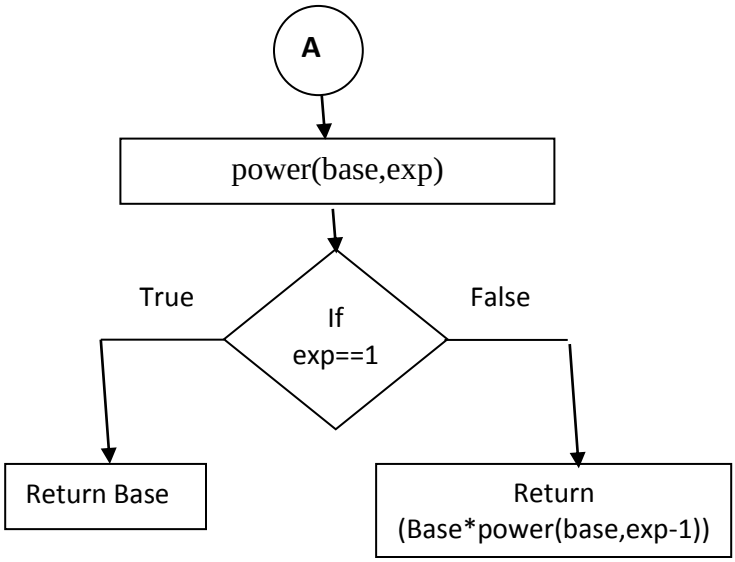
To write a python program to find exponentiation (power of a Number)

Algorithm:

1. Create a function using def
2. Check if exponent value
    a. If exponent == 1
        Return base value
    b. else
        Recursively call the function with (base, exponent-1)
        Multiply the result returned by each recursive call with base value and Return the final result
3. Read base & exponent values using input function
4. Call the function & Print the Result

Flowchart:





Program:

```
def power(base,exp):
    if  exp==1:
        return(base)
    if  exp!=1:
        return(base*power(base,exp-1))
base=int(input("Enter base: "))
exp=int(input("Enter exponential value: "))
print("Result:",power(base,exp))
```

Output:

```
Enter base: 2
Enter exponential value: 2
Result: 4
```

Result:

Thus, the program to find exponentiation (power of a Number) is executed successfully

Ex. No 4a **FIND THE MAXIMUM OF A LIST OF NUMBERS**

Date:

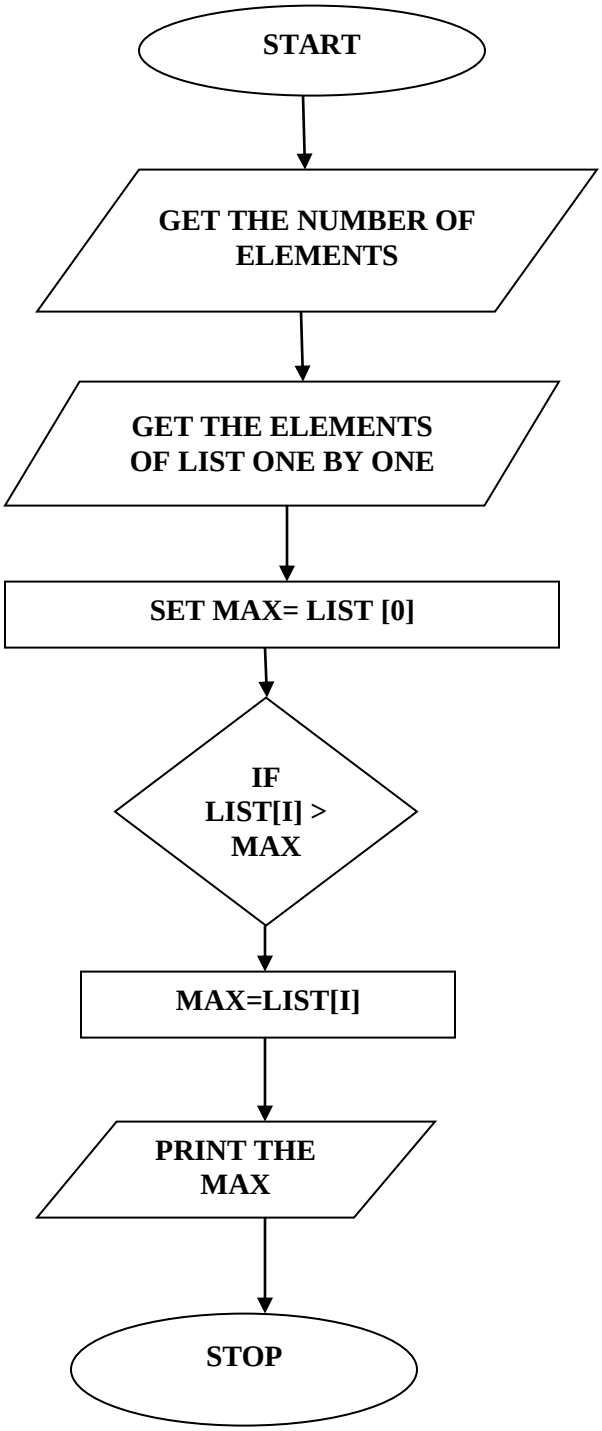
Aim:

To write a python program to find Maximum of a List of Numbers

Algorithm:

1. Initialize a List
2. Read the Number of Elements
3. Read the List values one by one
4. Set Max = first element of the list
5. Compare Max with List elements
6. If Max< List element
7. Set Max= List element
8. Continue the step 5 until reach the end of the List
9. Print the Max

Flowchart:


START
GET THE NUMBER OF ELEMENTS
GET THE ELEMENTS OF LIST ONE BY ONE
SET MAX= LIST [0]
IF LIST[I] > MAX
MAX=LIST[I]
PRINT THE MAX
STOP

Program :

```
def MaxList():
    list1=[]
    N=int(input("Enter Number of Values"))
    for i in range (0,N):
        x=int(input("Enter a Value"))
        list1.append(x)
    print(list1)
    Max=list1[0]
    j=1
    while j<N:
        if  list1[j]>Max :
            Max=list1[j]
    j=j+1
    print(" The Maximum Element in the List is ",Max)
MaxList()
```

Output:

```
Enter Number of Values: 3
Enter a Value56
Enter a Value76
Enter a Value43
 [56, 76, 43]
 The Maximum Element in the List is  76
```

Result:

Thus, the program to find Maximum of a List of Numbers is executed successfully

Ex. No 4b **REMOVING ALL THE DUPLICATE ELEMENTS IN A LIST**
Date:

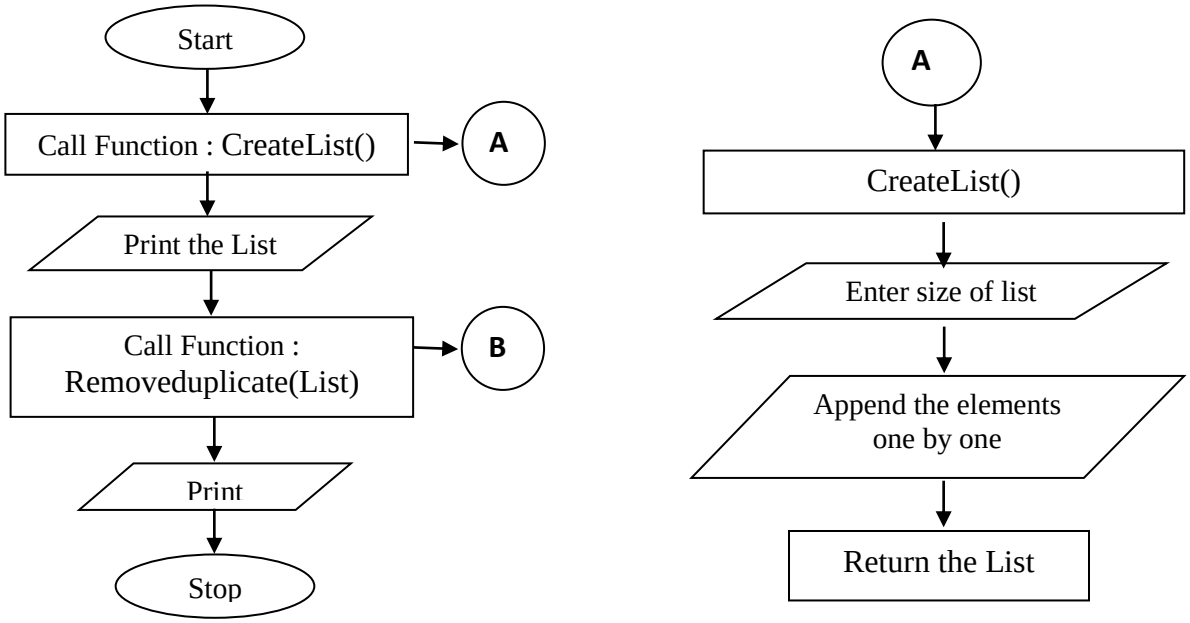
Aim:

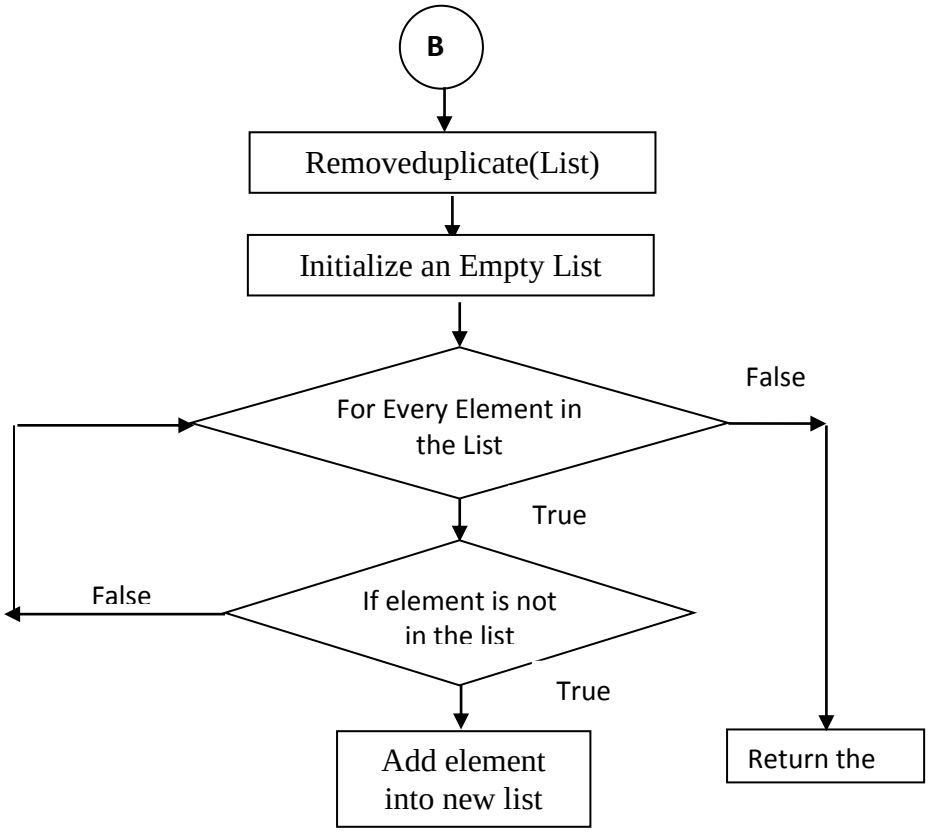
To write a python program to remove all duplicate elements in the list

Algorithm:

1. Cretae a list using creatlist()
   a. Enter number of elements
   b. Append the values one by one in the list
2. Print the list
3. Pass the list as input to removeduplicat() function
   a. Initialize an empty list
   b. Initialize an empty set (set only contains the unique elements used to remove the duplicates)
   c. For every element in the list,
      i. if the element is not in the set
         1. Add that element into both the list & set
      ii. Else repeat the step 3b until reach the end of the list
   d. Return this new list ,
4. Print the list after removing the duplicates

Flowchart:




B
Removeduplicate(List)
Initialize an Empty List
For Every Element in the List
False
True
If element is not in the list
False
True
Add element into new list
Return the

Program :

```
def createlist():
    list1=[]
    N=int(input("Enter Number of Values"))
    for i in range (0,N):
        x=int(input("Enter a Value"))
        list1.append(x)
    return list1
def removeduplicate(A):
    output=set()
    list2=[]
    for i in A:
        if i not in output:
            output.add(i)
            list2.append(i)
    return list2
A=createlist()
print(A)
result=removeduplicate(A)
print(result)
```

Output :

```
Enter Number of Values4
Enter a Value11
Enter a Value22
Enter a Value11
Enter a Value22
[11, 22, 11, 22 ]
[11, 22 ]
```

Result:

Thus, the program to remove all duplicate elements in the list is executed successfully

Ex. No 5a **LINEAR SEARCH**

Date:

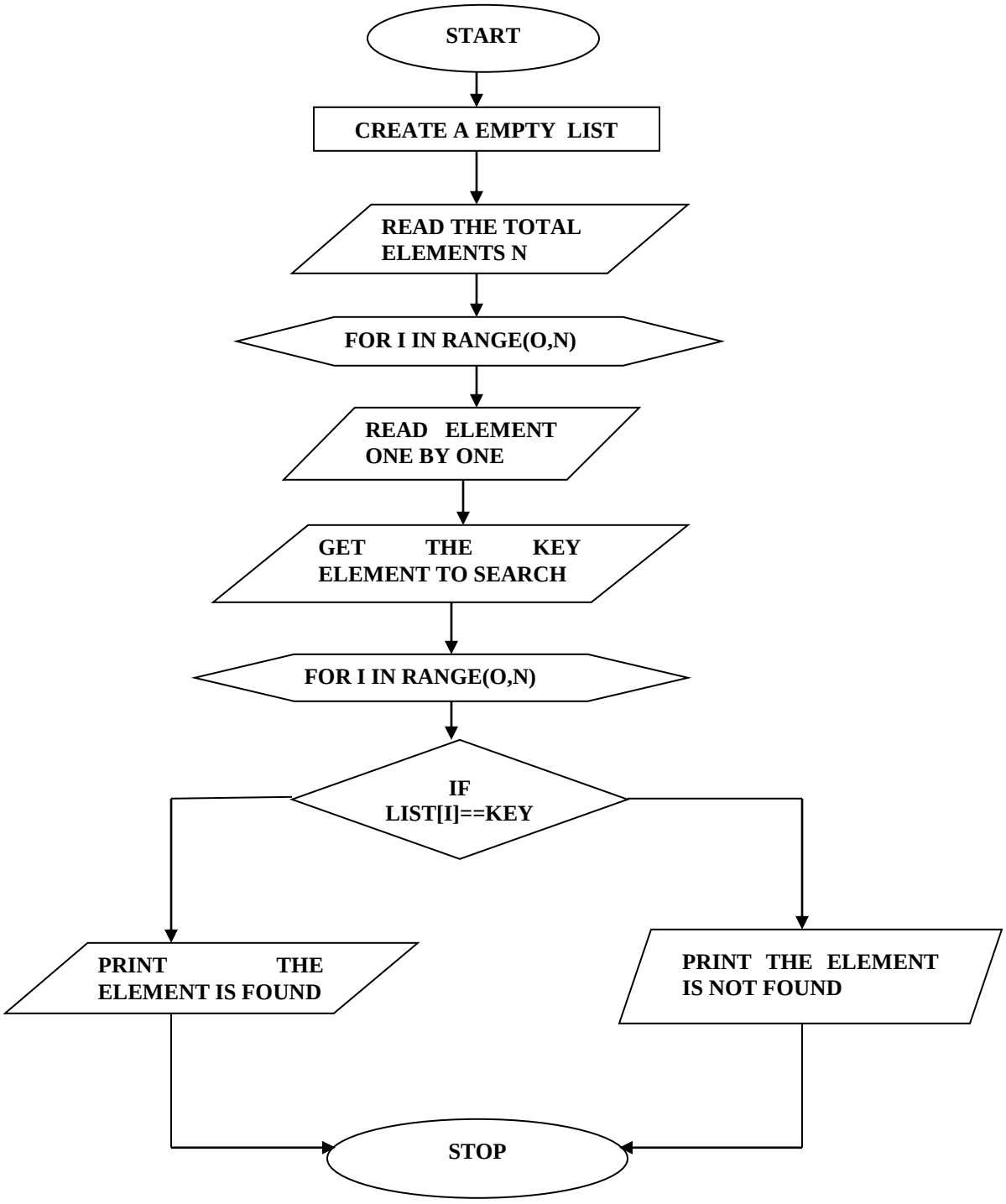
Aim :

To write a python program to perform Linear Search

Algorithm:

1. Define a function called LS()
2. Define a empty list Set a flag to 0
3. Read number of elements, store it in a variable called N
4. Read the values one by one using for loop
    a. Append each value into the List
5. Print the list
6. Read key values to be searched in the list
7. Check whether key present in the list one by one using loop statement
    a. if found ,Set flag to 1
8. If flag==1 , then print "*key* is found at position *pos_value*"
9. Else , print "key not found"
10. Call the function LS() at end of code

Flowchart:


START
CREATE A EMPTY LIST
READ THE TOTAL ELEMENTS N
FOR I IN RANGE(O,N)
READ ELEMENT ONE BY ONE
GET THE KEY ELEMENT TO SEARCH
FOR I IN RANGE(O,N)
IF LIST[I]==KEY
PRINT THE ELEMENT IS FOUND
PRINT THE ELEMENT IS NOT FOUND
STOP

Program:

```
def LS() :
        list1=[]
        flag=0
        N=int(input("enter no. of elements"))
        print("Enter values one by one")
        for  i in range(0,N):
                a=int(input("enter a value"))
                list1.append(a)
        print(list1)
        key=int(input("enter the Key value"))
        for i in range(0,N) :
                if key == list1[i]:
                        flag=1
                        break
        if  flag==1:
                print(key ,"  is found in the list1 at position  ",i+1 )
        else:
                print(key ,"  is not found in the list1")
LS()
```

Output:

```
enter no.of values4
Enter values one by one
enter value11
enter value22
enter value33
enter value44
[11, 22, 33, 44]
enter key value22
22 is present in the List at position 2
```

Result:

Thus, the Linear Search has been performed successfully

Ex. No 5b **BINARY SEARCH**

Date:

Aim:

To write a python program to perform Binary search

Algorithm:

1. Define a function called BS(alist,key)
2. Set a first & last values to point the boundary of list to be searched
3. Find mid position of the given list called alist
4. Compare key with mid position element
    a. If mid element == key

        Set found=True, print key founf

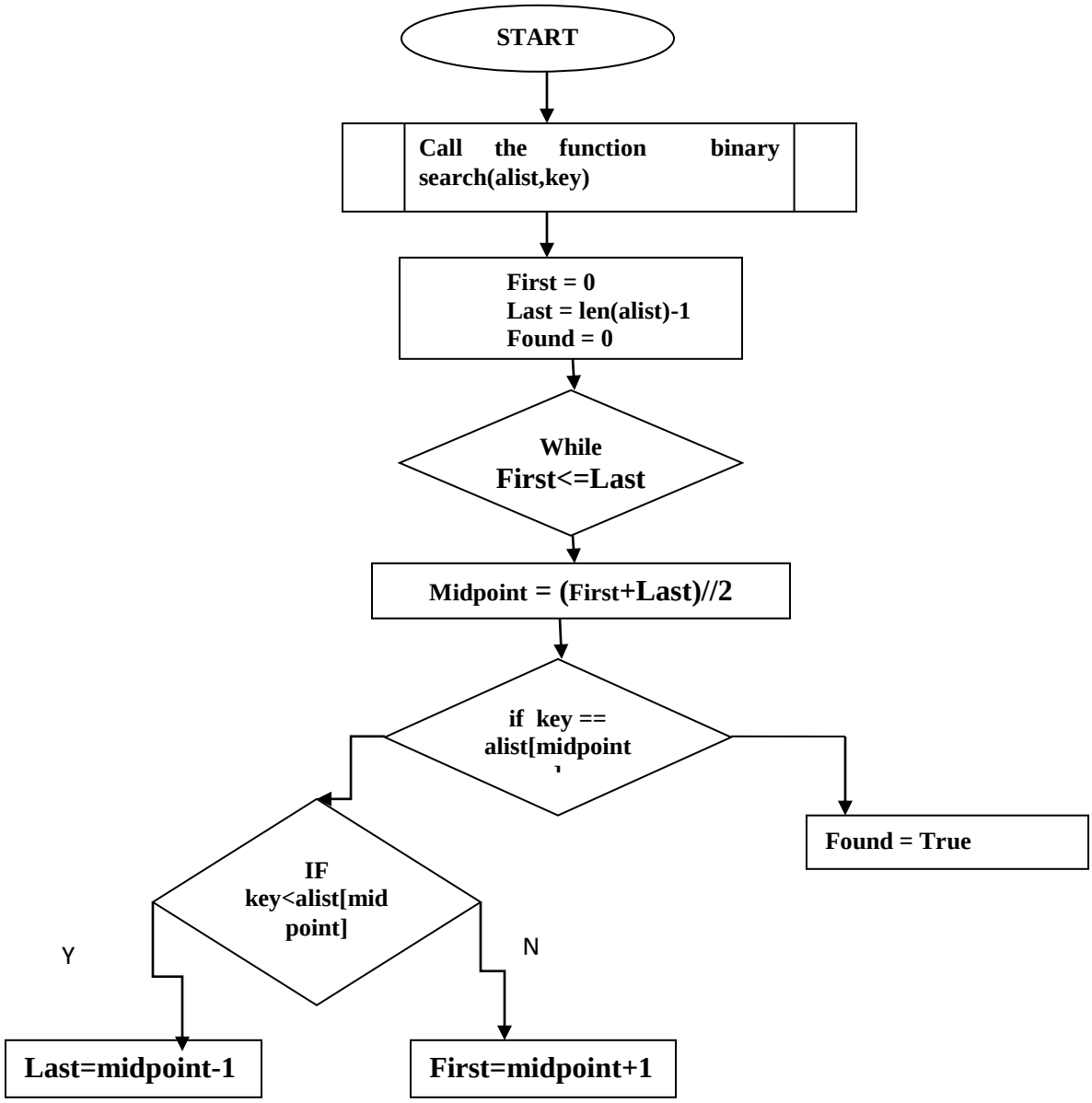
    b. If mid element < key

        Set last= midposition -1

    c. If mid element > key

        Set first= midposition +1
5. Continue the steps 3 & 4 until the element found or reach the end of the list
6. Initialize a list (elements in sorted order)
7. Call BS() with name of list & key value : BS(alist,key)
8. Print the result

Flowchart:


START
Call the function binary search(alist,key)
First = 0
Last = len(alist)-1
Found = 0
While First<=Last
Midpoint = (First+Last)//2
if key == alist[midpoint
Found = True
IF key<alist[mid point]
Y
N
Last=midpoint-1
First=midpoint+1

Program :

```
def  binarySearch(alist, key) :
        first = 0
        last = len(alist)-1
        found = 0
        while  first<=last  and  not  found:
                midpoint = (first+last)//2
                if  key==alist[midpoint] :
                        found=1
                        break
                elif key<alist[midpoint] :
                        last=midpoint-1
                else:
                        first=midpoint+1
        if found==1:
                print(key , "is found in the list " ,alist ,"at position   ",midpoint+1)
        else :
                print(key , "is not found in the list ")

testlist = [0, 1, 2, 8, 13, 17, 19, 32, 42,]
binarySearch(testlist, 13)
testlist = [0, 1, 2, 8, 13, 17, 19, 32, 42,]
binarySearch(testlist, 100)
```

Output:

```
13 is found in the list  [0, 1, 2, 8, 13, 17, 19, 32, 42] at position    5
100 is not found in the list
```

Result:

Thus, the Binary Search has been performed successfully

Ex. No 6a **SELECTION SORT**

Date:

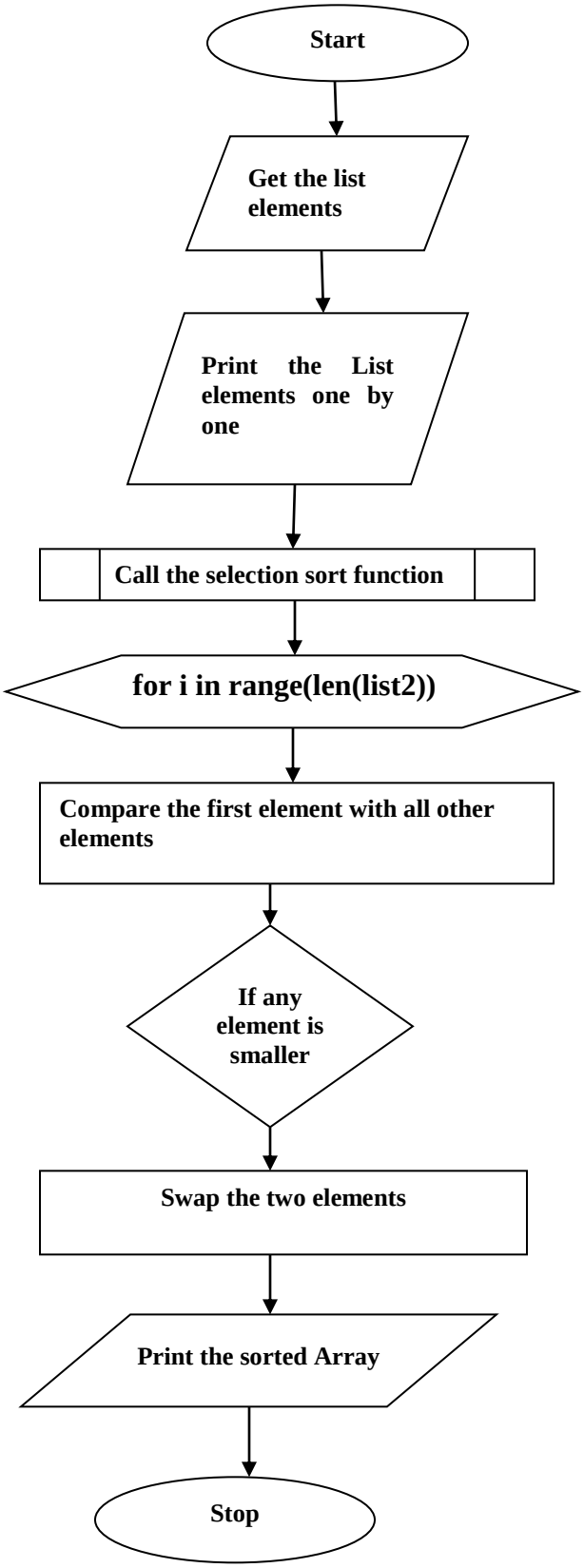
Aim:

To write a python program to perform selection sort

Algorithm:

1. Define a function selection()
2. Call this function by passing a list as input to this function
3. For every element in the list
    a. Pick the first element
    b. Compare it with the remaining elements of the list
    c. Find an element that is smaller than the first one
    d. Swap these two elements (smallest,first)
    e. Now, the smallest element placed at first position, Do the step 3 for next element
    f. Continue the step3 until all the elements are arranged in sorted order
4. Return the sorted list
5. Display the sorted list

Flowchart:


Start
Get the list elements
Print the List elements one by one
Call the selection sort function
for i in range(len(list2))
Compare the first element with all other elements
If any element is smaller
Swap the two elements
Print the sorted Array
Stop

Program:

```
def selection(list2):
        for i in range(len(list2)):
                least=i
                for k in range(i+1,len(list2)):
                        if list2[k] < list2[least]:
                                least=k
                list2=swap (list2,least,i)
        return list2

def swap(A,x,y):
        tmp=A[x]
        A[x]=A[y]
        A[y]=tmp
        return A

list1=[25,9,8,3,5,7,10]
print("Before Sorting  " ,list1)
result=selection(list1)
print("After Sorting  " ,result)
```

Output:

Before Sorting [25, 9, 8, 3, 5, 7, 10]
After Sorting [3, 5, 7, 8, 9, 10, 25]

Result:

Thus, the program for selection sort has been executed successfully

Ex. No 6b **INSERTION SORT**

Date:

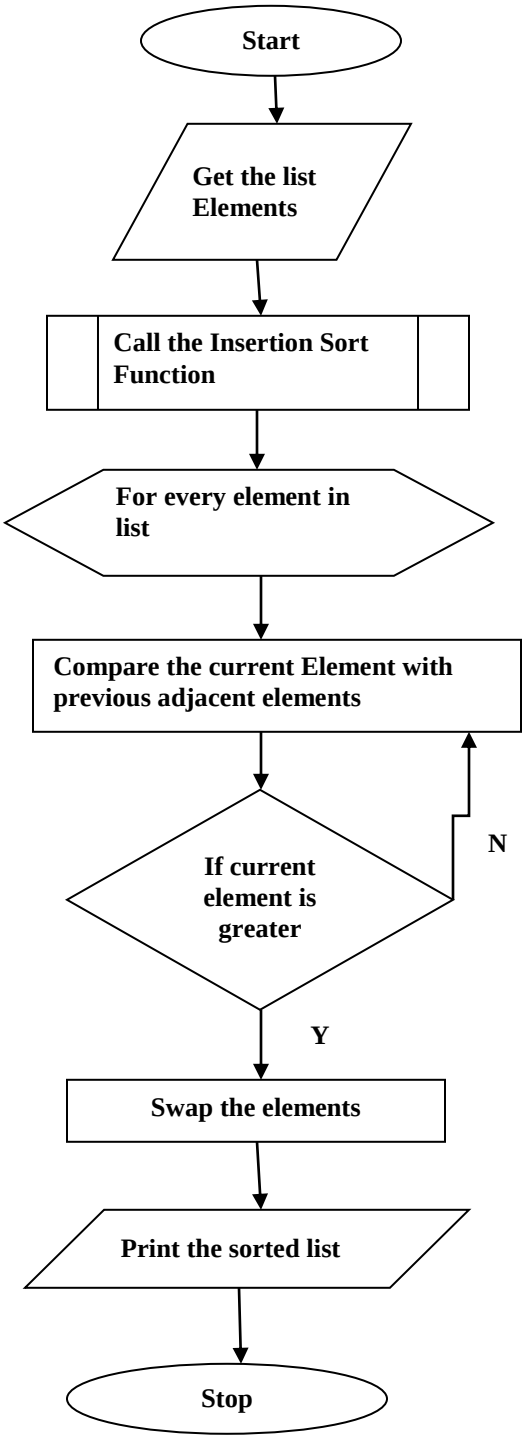
Aim:

To write a python program to perform Insertion sort

Algorithm:

1. Define a function Insertion()
    a. For every element in the list
        i. Set current element
        ii. Compare the current element with its previous adjacent element
            If current element < previous adjacent element
                Swap these two elements
            Continue the step (ii) until all the predecessors to the current element are arranged in order
        iii. Repeat the step 1a until all elements in the list are arranged in sorted order
    b. Print the sorted list
2. Get a List
3. Call the function by passing the list as input
4. Display the result

Flowchart:


Start
Get the list Elements
Call the Insertion Sort Function
For every element in list
Compare the current Element with previous adjacent elements
If current element is greater
N
Y
Swap the elements
Print the sorted list
Stop

Program:

```
def insertion(a):
        for  i  in  a :
                j = a.index(i)
                while  j>0 :
                        if  a[j-1]  >  a[j] :
                                a[j-1],a[j] = a[j],a[j-1]
                        else:
                                break
                        j = j-1
        print("After Sorting   : ",a)

list1 = [16,19,11,15,10,12,14,5]
print("Before Sorting  :",list1)
insertion(list1)
```

Output:

Before Sorting: [16, 19, 11, 15, 10, 12, 14, 5]

After Sorting: [5, 10, 11, 12, 14, 15, 16, 19]

Result:

Thus, the program for Insertion sort has been executed successfully

Ex. No 7 **MERGE SORT**

Date:

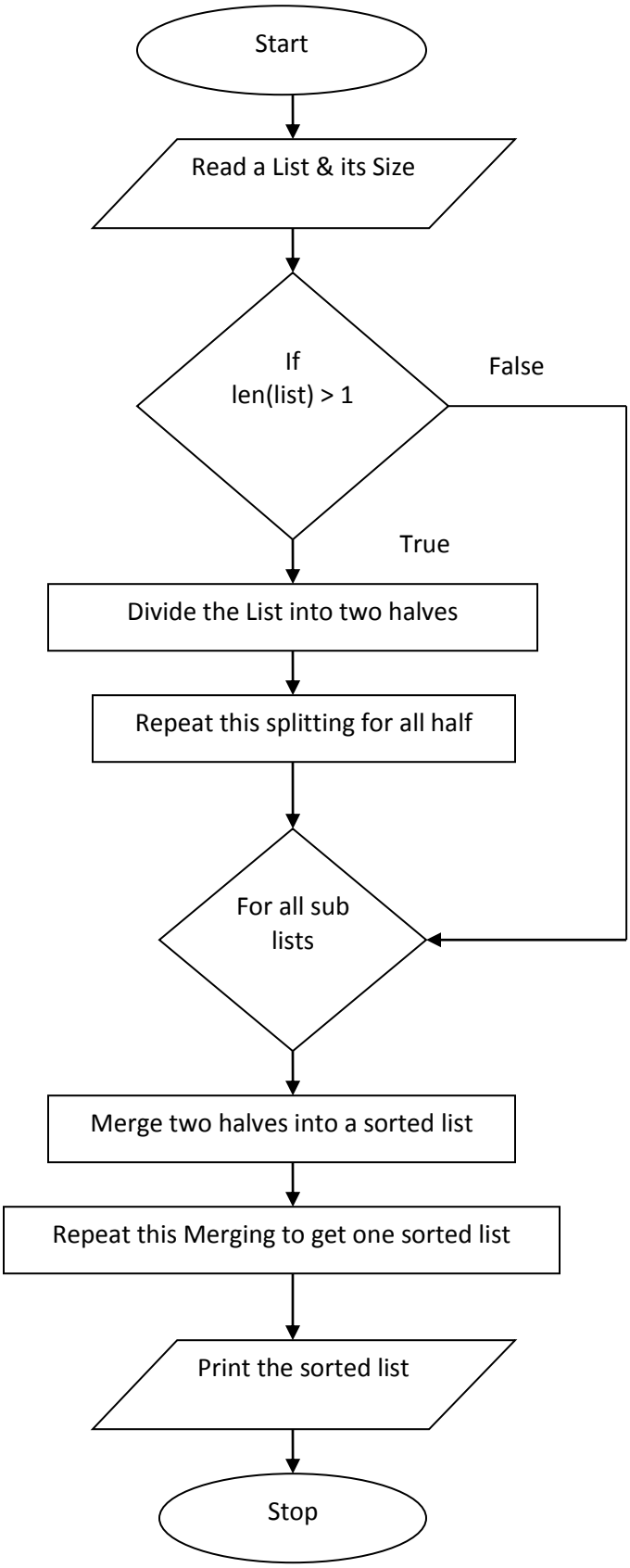
Aim:

To write a python program to perform Merge sort

Algorithm:

1. Create copies of the subarrays L ← A[p..q] and M ← A[q+1..r].
2. Create three pointers i,j and k
    1. i maintains current index of L, starting at 1
    2. j maintains current index of M, starting at 1
    3. k maintains current index of A[p..q], starting at p
3. Until we reach the end of either L or M, pick the larger among the elements from L and M and place them in the correct position at A[p..q]
4. When we run out of elements in either L or M, pick up the remaining elements and put in A[p..q]

Flowchart:


Start
Read a List & its Size
If
len(list) > 1
False
True
Divide the List into two halves
Repeat this splitting for all half
For all sub
lists
Merge two halves into a sorted list
Repeat this Merging to get one sorted list
Print the sorted list
Stop

Program:

```
def mergeSort(alist):
    print("Splitting ",alist)
    if len(alist)>1:
        mid = len(alist)//2
        lefthalf = alist[:mid]
        righthalf = alist[mid:]
        mergeSort(lefthalf)
        mergeSort(righthalf)
        i=0
        j=0
        k=0
        while i < len(lefthalf) and j < len(righthalf):
            if lefthalf[i] < righthalf[j]:
                alist[k]=lefthalf[i]
                i=i+1
            else:
                alist[k]=righthalf[j]
                j=j+1
            k=k+1
        while i < len(lefthalf):
            alist[k]=lefthalf[i]
            i=i+1
            k=k+1
        while j < len(righthalf):
            alist[k]=righthalf[j]
            j=j+1
            k=k+1
    print("Merging ",alist)
alist = [54,26,93,17,77,31,44,55,20]
mergeSort(alist)
print(alist)
```

Output:

('Splitting ', [54, 26, 93, 17, 77, 31, 44, 55, 20])

('Splitting ', [54, 26, 93, 17])

('Splitting ', [54, 26])

('Splitting ', [54])

('Merging ', [54])

('Splitting ', [26])

('Merging ', [26])

('Merging ', [26, 54])

('Splitting ', [93, 17])

('Splitting ', [93])

('Merging ', [93])

('Splitting ', [17])

('Merging ', [17])

('Merging ', [17, 93])

('Merging ', [17, 26, 54, 93])

('Splitting ', [77, 31, 44, 55, 20])

('Splitting ', [77, 31])

('Splitting ', [77])

('Merging ', [77])

('Splitting ', [31])

('Merging ', [31])

('Merging ', [31, 77])

('Splitting ', [44, 55, 20])

('Splitting ', [44])

('Merging ', [44])

('Splitting ', [55, 20])

('Splitting ', [55])

('Merging ', [55])

('Splitting ', [20])

('Merging ', [20])

('Merging ', [20, 55])

('Merging ', [20, 44, 55])

('Merging ', [20, 31, 44, 55, 77])

('Merging ', [17, 20, 26, 31, 44, 54, 55, 77, 93])

[17, 20, 26, 31, 44, 54, 55, 77, 93]

Result:

Thus, the program for merge sort has been executed successfully

Ex. No 8 **FIRST N PRIME NUMBERS**

Date:

Aim:

To write a python program to find first N prime Numbers

Algorithm:

1. Read lower & Upper bound values for the Range
2. Foe each number in the Range
3. Divide the num by 2 to num-1
4. Check the remainder

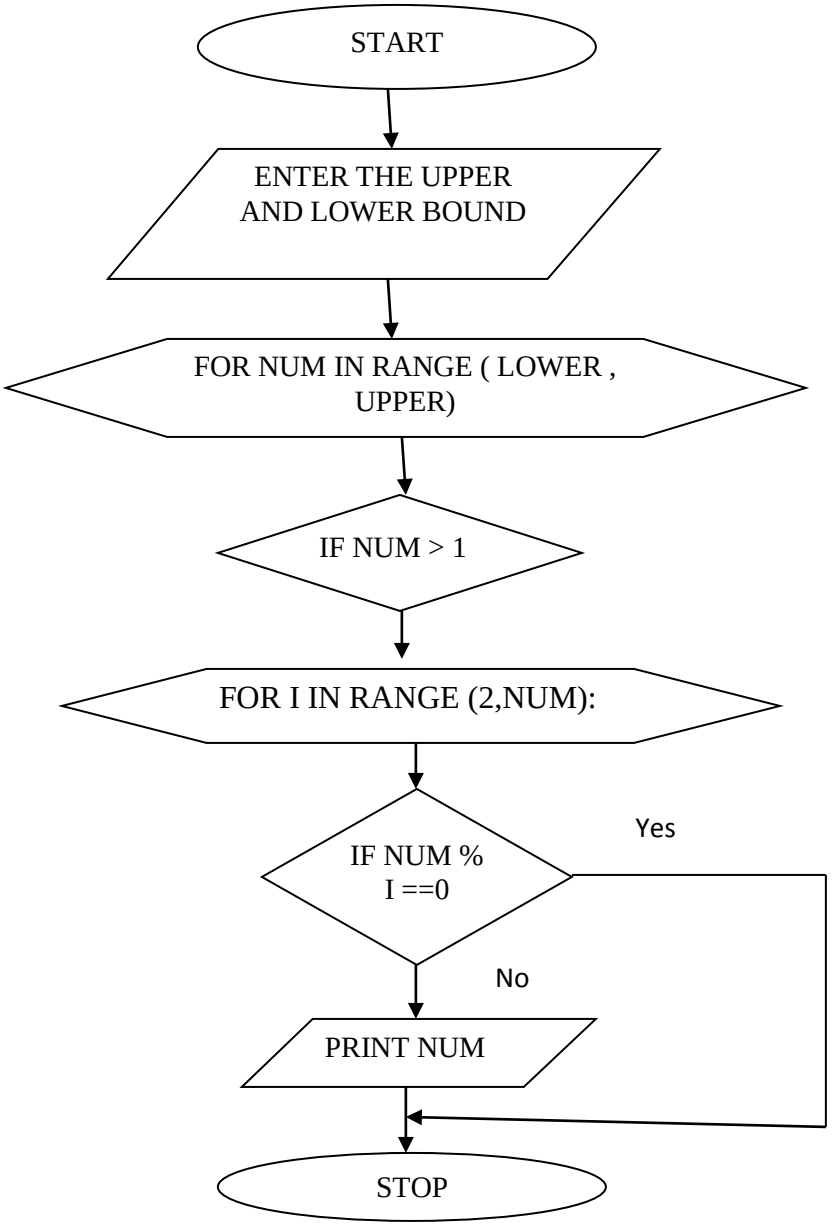
   If Remainder == 0 then,

   Num is not a prime number.

   Else

   Print the Number as Prime
5. Repeat the Steps 2 to 4 until reach the Upper bound of the Range

Flowchart:


START
ENTER THE UPPER AND LOWER BOUND
FOR NUM IN RANGE (LOWER , UPPER)
IF NUM > 1
FOR I IN RANGE (2,NUM):
IF NUM % I ==0
Yes
No
PRINT NUM
STOP

Program :

```
lower = int(input("Enter lower range: "))
upper = int(input("Enter upper range: "))
print("Prime numbers between",lower,"and",upper,"are:")
for  num in range(lower,upper + 1):
        if num > 1:
                for i in range(2,num):
                        if (num%i) == 0:
                                break
                else:
                        print(num)
```

Output:

```
Enter lower range: 2
Enter upper range: 10
Prime numbers between 1 and 10 are:
2
3
5
7
```

Result:

Thus, the program to find first N prime Numbers has been executed successfully

Ex. No 9 **MULTIPLICATION OF TWO MATRICES**
Date:

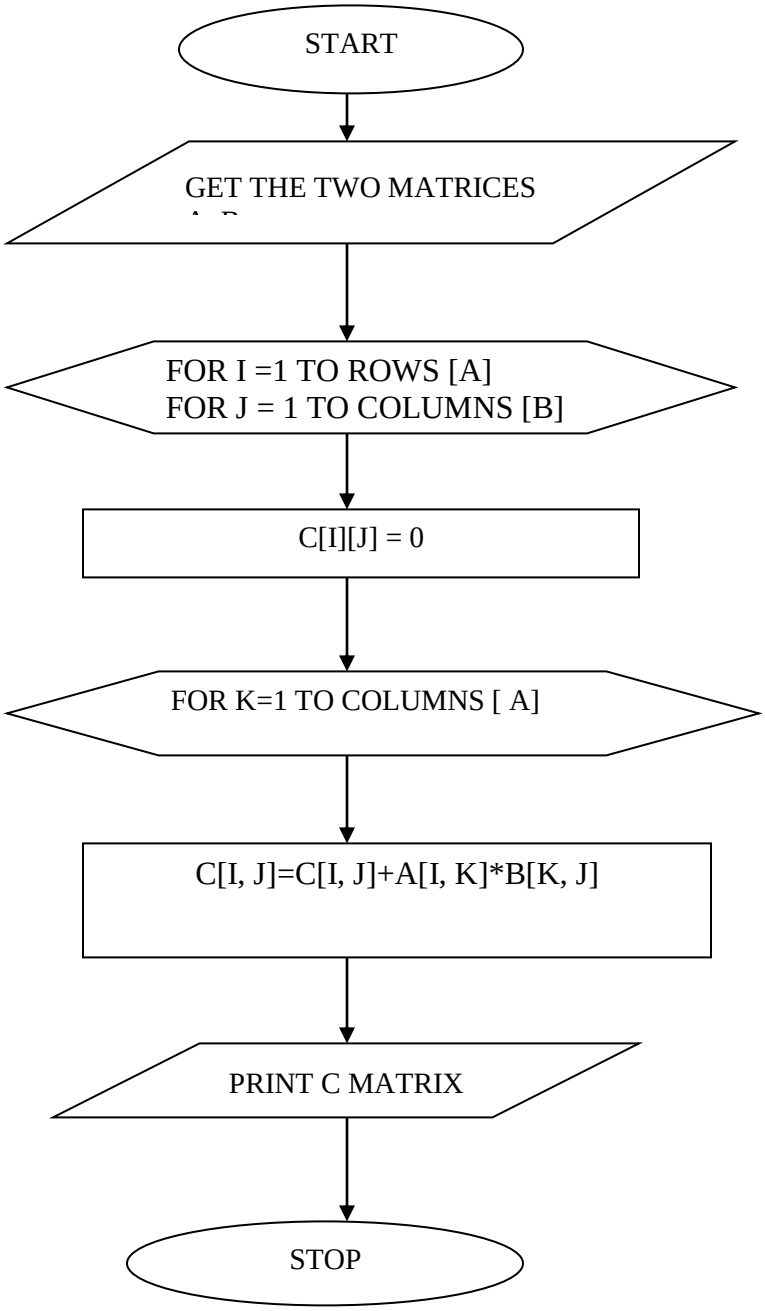
Aim:

To write a python program to multiply two matrices

Algorithm:

1. Create two lists with nested index
2. Initialize an empty list
3. Multiply two matrices
4. Store the results into empty list
5. Display the result

Flowchart:


START
GET THE TWO MATRICES
FOR I =1 TO ROWS [A]
FOR J = 1 TO COLUMNS [B]
C[I][J] = 0
FOR K=1 TO COLUMNS [A]
C[I, J]=C[I, J]+A[I, K]*B[K, J]
PRINT C MATRIX
STOP

Program:

```
matrix1 = [[1, 2, 3],[1, 2, 3],[1, 2, 3]]
matrix2 = [[1, 1, 1, 1], [1, 1, 1, 1],[1, 1, 1, 1]]
rmatrix = [[0, 0, 0, 0],[0, 0, 0, 0],[0, 0, 0, 0]]
for i in range(len(matrix1)):
        for j in range(len(matrix2[0])):
                for k in range(len(matrix2)):
                        rmatrix[i][j] += matrix1[i][k] * matrix2[k][j]


for r in rmatrix:
        print(r)
```

Output:

```
[6, 6, 6, 6]
[6, 6, 6, 6]
[6, 6, 6, 6]
```

Result:

Thus, the program to multiply two matrices has been executed successfully

Ex. No 10 **FIND THE MOST FREQUENT WORDS IN A TEXT READ FROM A FILE**
Date:

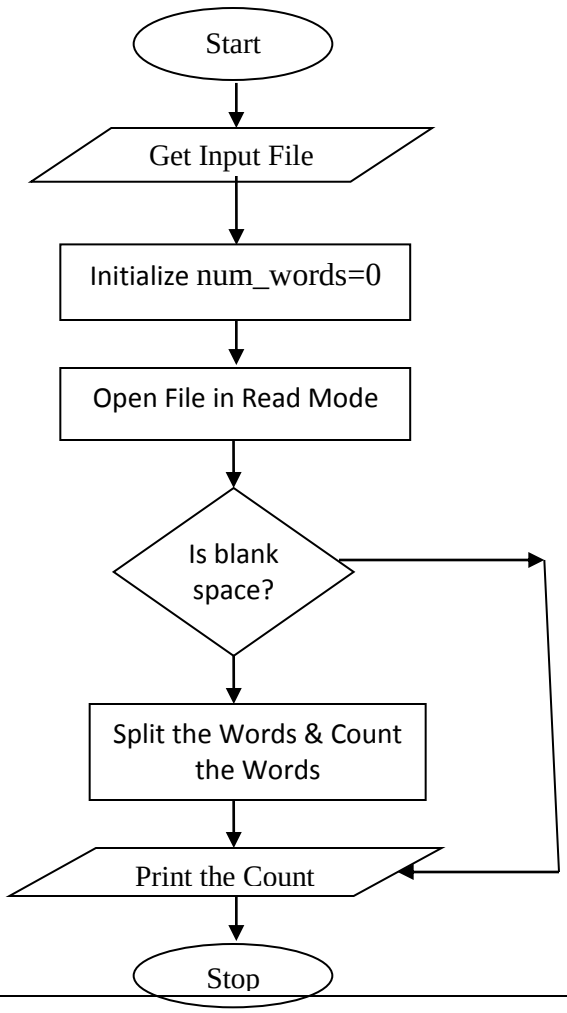
Aim:

Write a python program to find the most frequent words in a text read from a file

Algorithm:

1. Get the input file to be read.
2. Initialize num_words to zero.
3. Open the input file in read mode.
4. Use for loop to check the blank space, if there is a space then split the words and counts the words.
5. Repeat the step 4 until reach end of file.
6. Print the number of words.

Flowchart:



Program:

```
fname = input("Enter file name: ")
num_words = 0
with open(fname, 'r') as f:
    for line in f:
        words = line.split()
        num_words += len(words)
print("Number of words:")
print(num_words)
```

To create a text file : file->new->type sentences->save it as data1.txt

Hai Welcome to all

Output:

```
Enter file name: data1.txt
Number of words:
4
```

Result:

Thus, the program to find the most frequent words in a text read from a file has been executed successfully

Ex. No 11 **WORD COUNT USING COMMAND LINE ARGUMENTS**
Date:

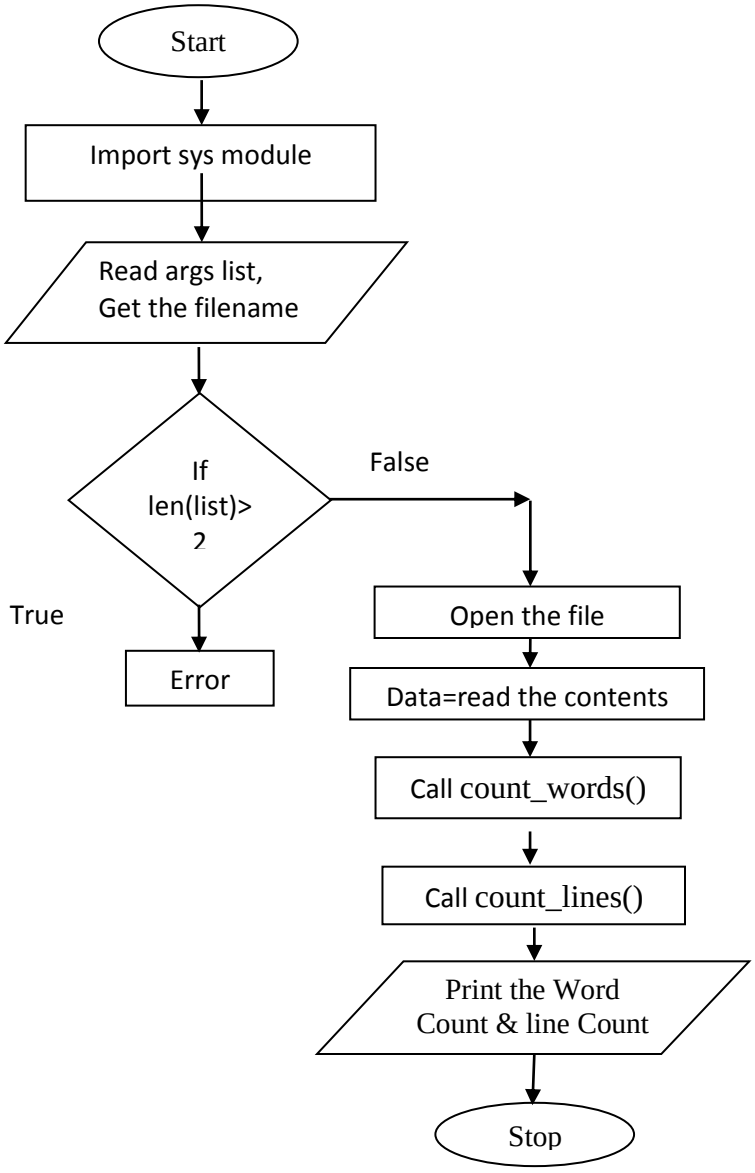
Aim:

To write a python program to implement word count using command line arguments

Algorithm:

1. Import the Sys module.
2. If less than 2 arguments are given, error will be generated.
3. Define functions count_words and count_lines.
4. Split the characters in the file into words and count the number of words.
5. Split the words if new lines is found and count the number of lines.
6. Return the number of words and number of lines in the file.

Flowchart:


Start
Import sys module
Read args list,
Get the filename
If
len(list)>
2
False
True
Error
Open the file
Data=read the contents
Call count_words()
Call count_lines()
Print the Word
Count & line Count
Stop

Program:

```
#e11.py
        import sys
        if len(sys.argv) < 2:
              print("Usage: python word_count.py <file>")
              exit(1)

        def count_words(data):
                words = data.split(" ")
                num_words = len(words)
                return num_words
        def count_lines(data):
                lines = data.split("\n")
                for l in lines:
                        if not l:
                                lines.remove(l)
                 return len(lines)

        filename = sys.argv[1]
        f = open(filename, "r")
        data = f.read()
        f.close()
        num_words = count_words(data)
        num_lines = count_lines(data)
        print("The number of words: ", num_words+num_lines-1)
        print("The number of lines: ", num_lines)
```

To create a text file : file->new->type sentences->save it as data1.txt

Hai Welcome to all
Python Programming Language

Output:

```
C:\python27> e11.py    data1.txt
Enter file name: data1.txt
The number of words: 7
The number of lines: 2
```

Result:

Thus, the program to implement word count using command line arguments has been executed successfully.

Ex. No 12 **SIMULATE ELLIPTICAL ORBITS IN PYGAME**
Date:

Aim:

Write a python program to simulate elliptical orbits in pygame

Algorithm:

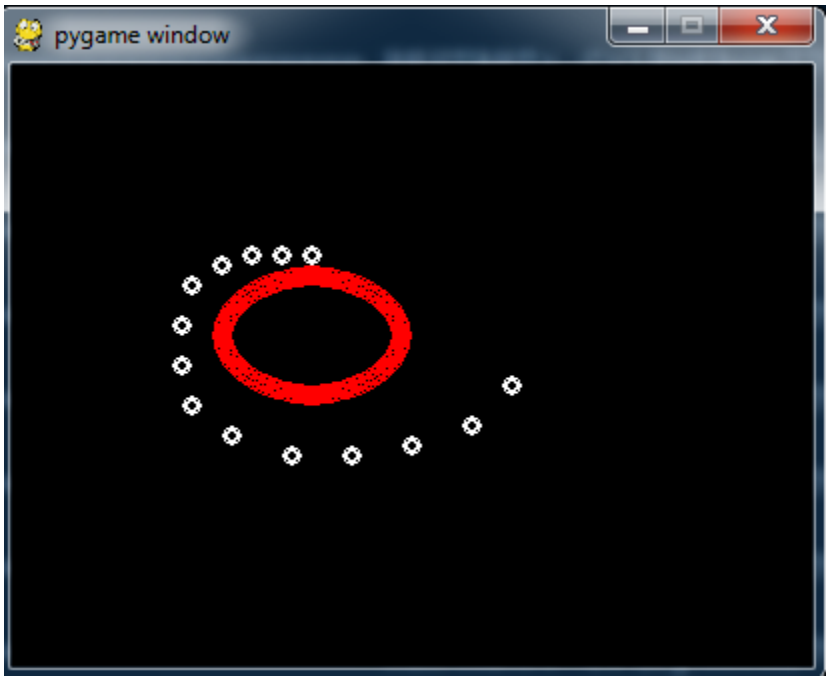
1. Import pygame module
2. Call pygame.init() to initiate all imported pygame module
3. Set the screen size in terms of pixels using pygame.display.set_mode((400, 300)
4. If there is any event in pygame queue
    a. Get the event from the pygame queue
    b. If event types is pygame.QUIT then set done=true
5. Else, Draw the circle & ellipse to display orbit
6. Call flip() method to update the full display Surface to the screen

Program:

```
import pygame
pygame.init()
screen = pygame.display.set_mode((400, 300))
done = False
while not done:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True
    pygame.draw.circle(screen, (255,255,255), [150, 95], 5, 3)
    pygame.display.update()
    pygame.draw.circle(screen, (255,255,255), [135, 95], 5, 3)
    pygame.draw.circle(screen, (255,255,255), [120, 95], 5, 3)
    pygame.draw.circle(screen, (255,255,255), [105, 100], 5, 3)
    pygame.draw.circle(screen, (255,255,255), [90, 110], 5, 3)
```

```
pygame.draw.circle(screen, (255,255,255), [85, 130], 5, 3)
pygame.draw.circle(screen, (255,255,255), [85, 150], 5, 3)
pygame.draw.circle(screen, (255,255,255), [90, 170], 5, 3)
pygame.draw.circle(screen, (255,255,255), [110, 185], 5, 3)
pygame.draw.circle(screen, (255,255,255), [140, 195], 5, 3)
pygame.draw.circle(screen, (255,255,255), [170, 195], 5, 3)
pygame.draw.circle(screen, (255,255,255), [200, 190], 5, 3)
pygame.draw.circle(screen, (255,255,255), [230, 180], 5, 3)
pygame.draw.circle(screen, (255,255,255), [250, 160], 5, 3)
pygame.draw.ellipse(screen, (255,0,0), [100, 100, 100, 70], 10)
pygame.display.flip()
```

Output:



Result:

Thus, the program to simulate elliptical orbits in pygame has been executed successfully.

Ex. No 13 **SIMULATE BOUNCING BALL USING PYGAME**
Date:

Aim:

Write a python program to simulate bouncing ball using pygame

Algorithm:

1. Import pygame module
2. Call pygame.init() to initiate all imported pygame module
3. Set the screen size in terms of pixels using pygame.display.set_mode((400, 300)
4. If there is any event in pygame queue
    a. Get the event from the pygame queue
    b. If event types is pygame.QUIT then set done=true
5. Else, Draw the circle update the screen display with new circle to bring bouncing effect
6. Call sys.exit() to uninitialized all the pygame modules

Program:

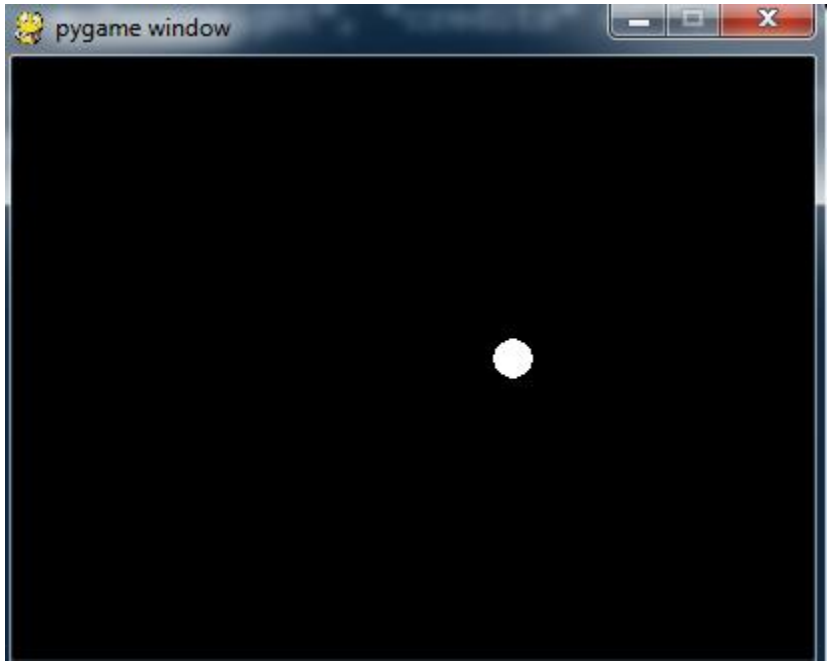
```
import pygame
from pygame.locals import *

pygame.init()
screen = pygame.display.set_mode((400, 300))
done = False

while not done:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            done = True
    pygame.draw.circle(screen, (255,255,255), [100, 80], 10, 0)
    pygame.display.update()
    pygame.draw.circle(screen, (0,0,0), [100, 80], 10, 0)
    pygame.display.update()
    pygame.draw.circle(screen, (255,255,255), [150, 95], 10, 0)
    pygame.display.update()
    pygame.draw.circle(screen, (0,0,0), [150, 95], 10, 0)
    pygame.display.update()
    pygame.draw.circle(screen, (255,255,255), [200, 130], 10, 0)
    pygame.display.update()
```

```
pygame.draw.circle(screen, (0,0,0), [200, 130], 10, 0)
pygame.display.update()
pygame.draw.circle(screen, (255,255,255), [250, 150], 10, 0)
pygame.display.update()
pygame.display.update()
for event in pygame.event.get():
   if event.type == QUIT:
      pygame.quit()
      sys.exit()
```

Output:



Result:

Thus, the program to simulate elliptical orbits in pygame has been executed successfully.