NRI INSTITUTE OF INFORMATION SCIENCE & TECHNOLOGY BHOPAL

DEPARTMENT OF INFORMATION TECHNOLOGY

LAB MANUAL

PYTHON

(IT – 605)

BACHELOR OF TECHNOLOGY

	NRI INSTITUTE OF INFORMATION SCIENCE & TECHNOLOGY **DEPT NAME : Information Technology**		**FORM NO**	**NIIST/A/10**
NIIST BHOPAL				
BRANCH	**IT**	**LIST OF EXPERIMENT**	**REV. NO**	**0**
SEMESTER	**VI**		**REV. DT**	**30/06/2011**
SUBJECT/ CODE : Python/IT-605				

S.NO.	LIST OF EXPERIMENT
1	To write a Python program to find GCD of two numbers
2	To write a Python Program to find the square root of a number by Newton's Method.
3	To write a Python program to find the exponentiation of a number
4	To write a Python Program to find the maximum from a list of numbers.
5	To write a Python Program to perform Linear Search
6	To write a Python Program to perform binary search.
7	To write a Python Program to perform selection sort.
8	To write a Python Program to perform insertion sort.
9	To write a Python Program to perform Merge sort.
10	To write a Python program to find first n prime numbers.
11	To write a Python program to multiply matrices.
12	To write a Python program for command line arguments.
13	To write a Python program to find the most frequent words in a text read from a file.
14	To write a Python program to simulate elliptical orbit sin Pyramid.
15	To write a Python program to bouncing ball in Pyramid

EXPERIMENT NO.1

AIM:

To write a Python program to find GCD of two numbers .

Algorithm:

1. Define a function named compute GCD()
2. Find the smallest among the two inputs x and y.
3. Perform the following step till smaller+1
   Check if ((x % i== 0) and(y% i== 0)), then assignGCD=i
4. Print the value of gcd

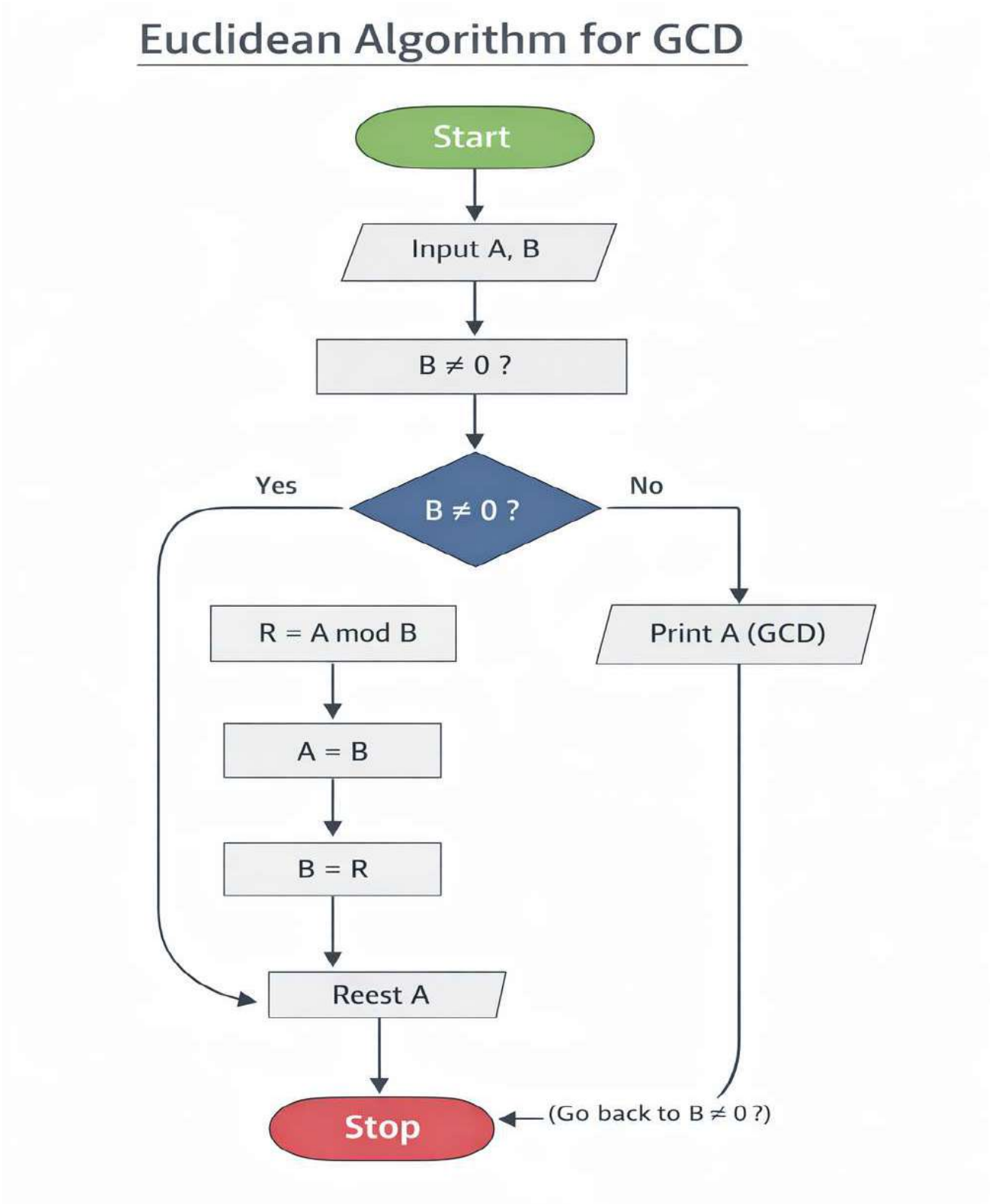
Program:

```
def computeGCD(x, y):
    # Find the smaller number
    if x > y:
        smaller = y
    else:
        smaller = x

    # Initialize gcd
    gcd = 1

    # Loop to find GCD
    for i in range(1, smaller + 1):
        if (x % i == 0) and (y % i == 0):
            gcd = i

    return gcd


# Given numbers
num1 = 54
num2 = 24

# Uncomment below lines to take input from user
# num1 = int(input("Enter first number: "))
# num2 = int(input("Enter second number: "))

print("The GCD of", num1, "and", num2, "is", computeGCD(num1, num2))
```

Output:

```
The GCD of 54 and 24 is 6
```

FLOWCHART :



EXPERIMENT NO.2

AIM:

To Write a Python Program to find the square root of a number by Newton's Method.

Algorithm:

1. Define a function named newtonSqrt().
2. Initialize approx as 0.5*n and better as 0.5*(approx +n/approx)
3. Use a while loop with a condition better!=approx. to perform the following.
   - Set approx.=better
   - Better = 0.5*(approx. + n /approx.)
4. Print the value of approx.

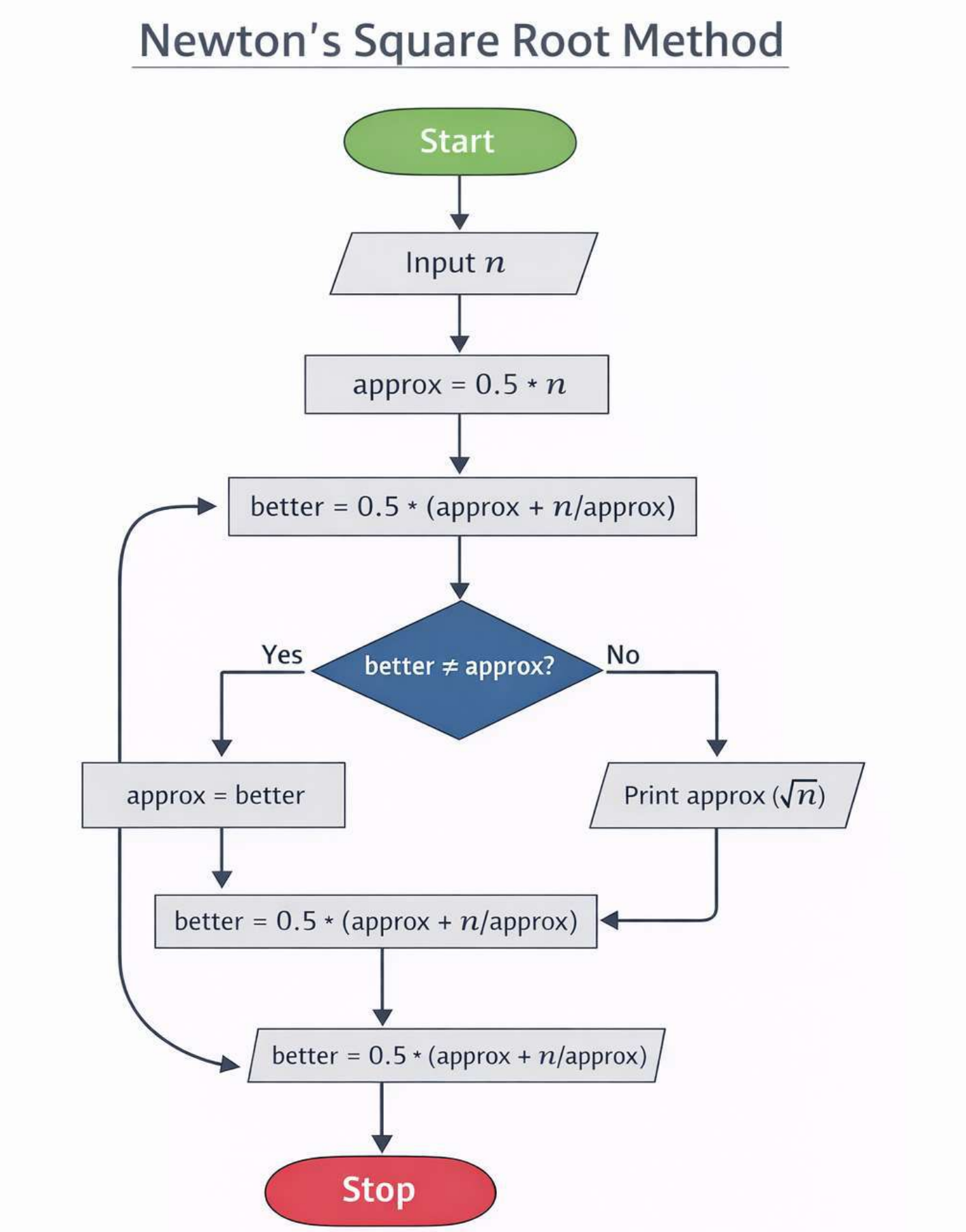
Program:

```
def newtonSqrt(n):
    approx = 0.5 * n
    better = 0.5 * (approx + n / approx)

    while better != approx:
        approx = better
        better = 0.5 * (approx + n / approx)

    return approx


# Example
print("The square root is", newtonSqrt(100))
```

Sample Output:

```
The square root is 10.0
```

Flowchart :


Newton's Square Root Method
Start
Input n
approx = 0.5 * n
better = 0.5 * (approx + n/approx)
better ≠ approx?
Yes
No
approx = better
Print approx ($\sqrt{n}$)
better = 0.5 * (approx + n/approx)
better = 0.5 * (approx + n/approx)
Stop

EXPERIMENT NO.3

AIM:

To write a Python program to find the exponentiation of a number.

Algorithm:

1. Define a function named power()
2. Read the values of base and exp
3. Use 'if' to check if exp is equal to 1 or not

i. If exp is equal to 1, then return base

ii. If exp is not equal to 1, then return (base*power(base,exp-1))

4. Print the result.

Program:

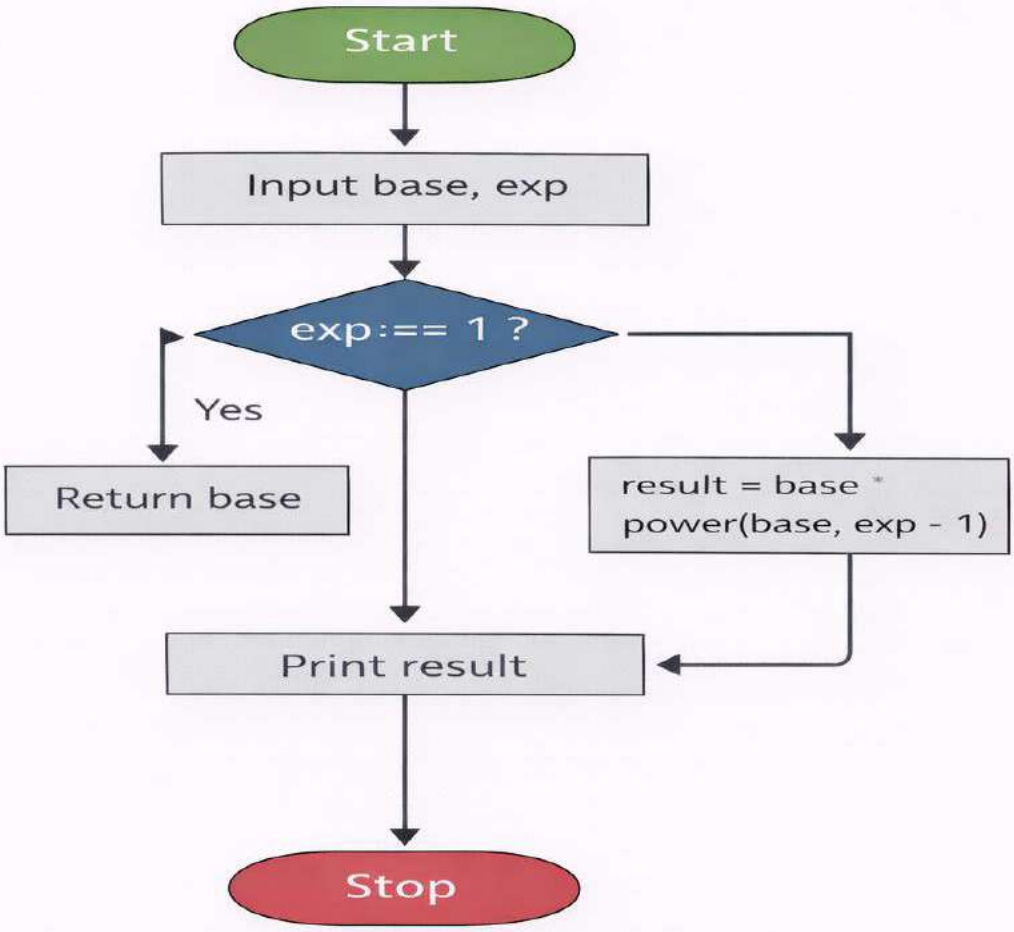
```
def power(base, exp):
    if exp == 1:
        return base
    else:
        return base * power(base, exp - 1)


base = int(input("Enter base: "))
exp = int(input("Enter exponential value: "))

print("Result:", power(base, exp))
```

Output :

```
 Enter base: 7
Enter exponential value: 2
Result: 49
```

Flowchart :

Power Calculation



EXPERIMENT NO.4

AIM:

To write a Python Program to find the maximum from a list of numbers.

Algorithm:

1. Create an empty list named l
2. Read the value of n
3. Read the elements of the list until n
4. Assign l [0]as max no
5. If l[i] > max no then set max no= l[i]
6. Increment by 1
7. Repeat steps 5-6 until i<n
8. Print the value of maximum number

Program:

```
l = []

n = int(input("Enter the upper limit: "))

for i in range(0, n):
    a = int(input("Enter the numbers: "))
    l.append(a)

maxno = l[0]

for i in range(0, len(l)):
    if l[i] > maxno:
        maxno = l[i]

print("The maximum number is %d" % maxno)
```

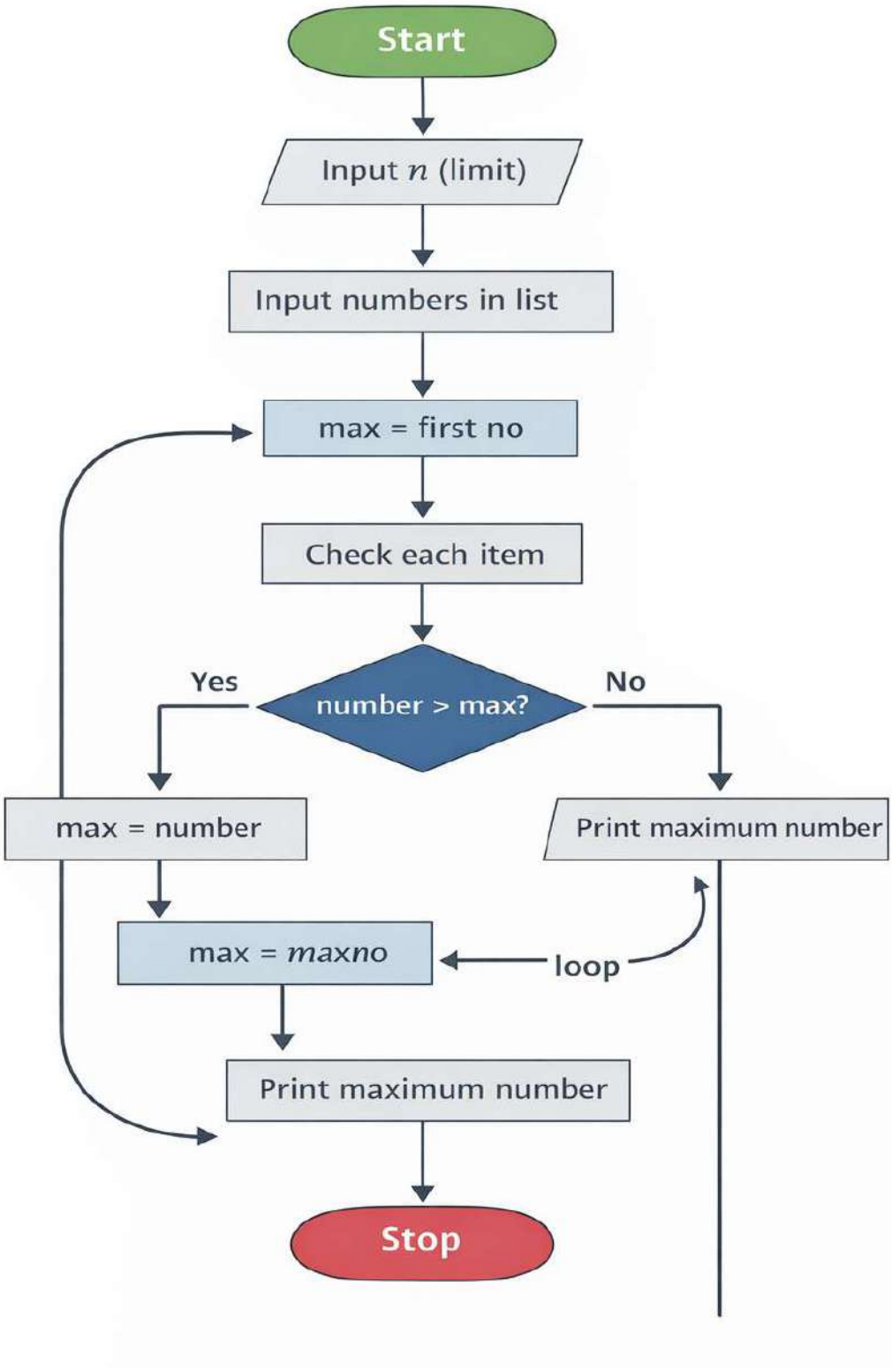
Sample Input:

```
 Enter the upper limit: 5
Enter the numbers: 10
Enter the numbers: 25
Enter the numbers: 7
Enter the numbers: 40
Enter the numbers: 12
```

Sample Output:

```
The maximum number is 40
```

Flowchart:



EXPERIMENT NO.5

AIM:

To write a Python Program to perform Linear Search

Algorithm:

1. Read n elements into the list
2. Read the element to be searched
3. If a list[pos]==item, then print the position n of the item
4. Else increment the position and repeat step3 until pos reaches the length of the list.

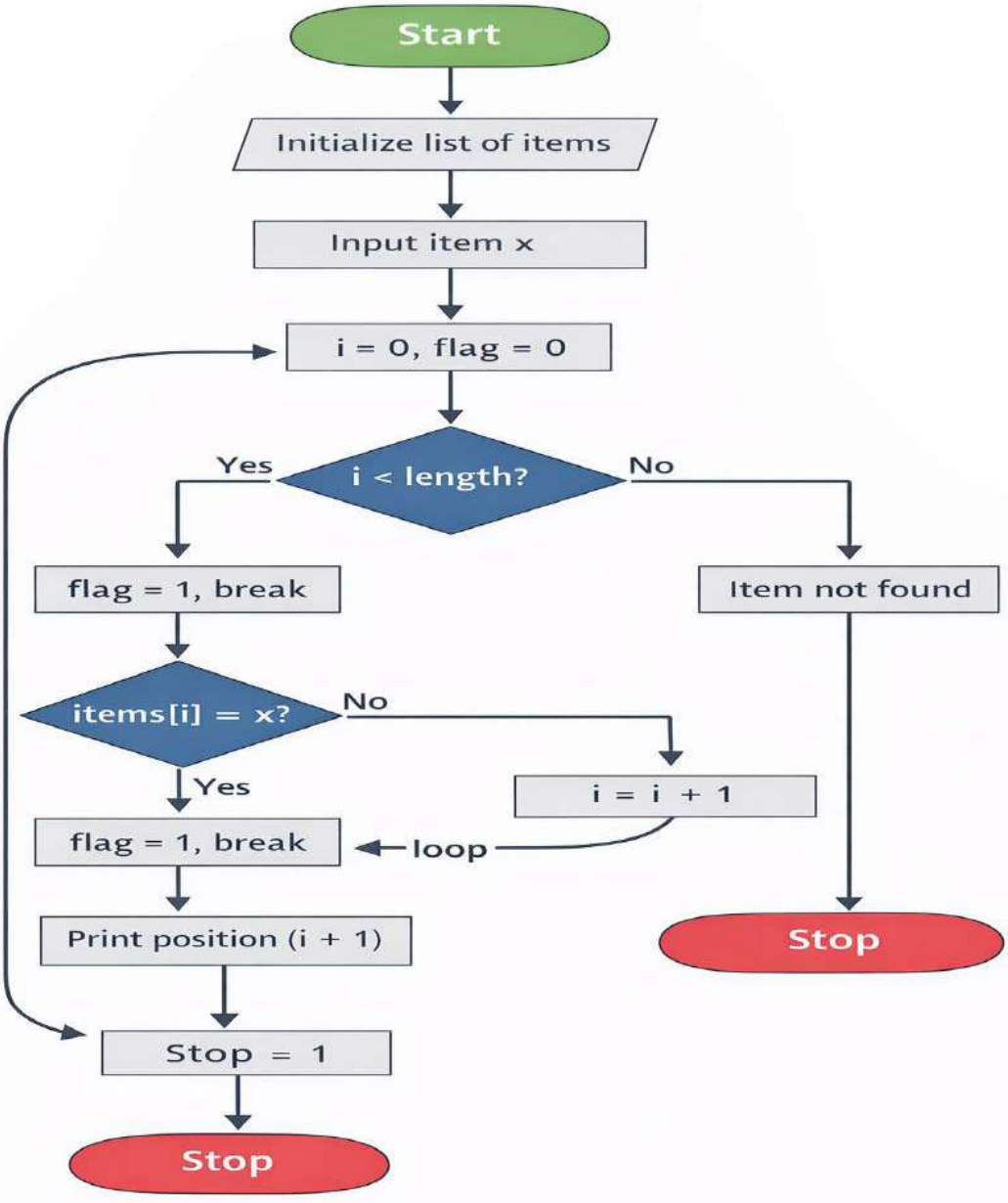
Program:

```
items = [5, 7, 10, 12, 15]

print("List of items is:", items)

x = int(input("Enter item to search: "))

i = 0
flag = 0

while i < len(items):
    if items[i] == x:
        flag = 1
        break
    i = i + 1

if flag == 1:
    print("Item found at position:", i + 1)
else:
    print("Item not found")
```

Sample Output:

```
 List of items is: [5, 7, 10, 12, 15]
Enter item to search: 7
Item found at position: 2
```

Flowchart :


Start
Initialize list of items
Input item x
i = 0, flag = 0
Yes
i < length?
No
flag = 1, break
Item not found
items[i] = x?
No
Yes
i = i + 1
flag = 1, break
loop
Print position (i + 1)
Stop
Stop = 1
Stop

EXPERIMENT NO.6

AIM:

To write a Python Program to perform binary search

Algorithm (Binary Search):

1. Start
2. Initialize a sorted list `arr`.
3. Set `low = 0` and `high = length of array - 1`.
4. Calculate `mid = (low + high) // 2`.
5. If `arr[mid] == x`, return `mid` (element found).
6. If `arr[mid] > x`, search in the **left subarray** by setting `high = mid - 1`.
7. If `arr[mid] < x`, search in the **right subarray** by setting `low = mid + 1`.
8. Repeat steps 4–7 until `low > high`.
9. If element is not found, return `-1`.
10. Stop.

Program :

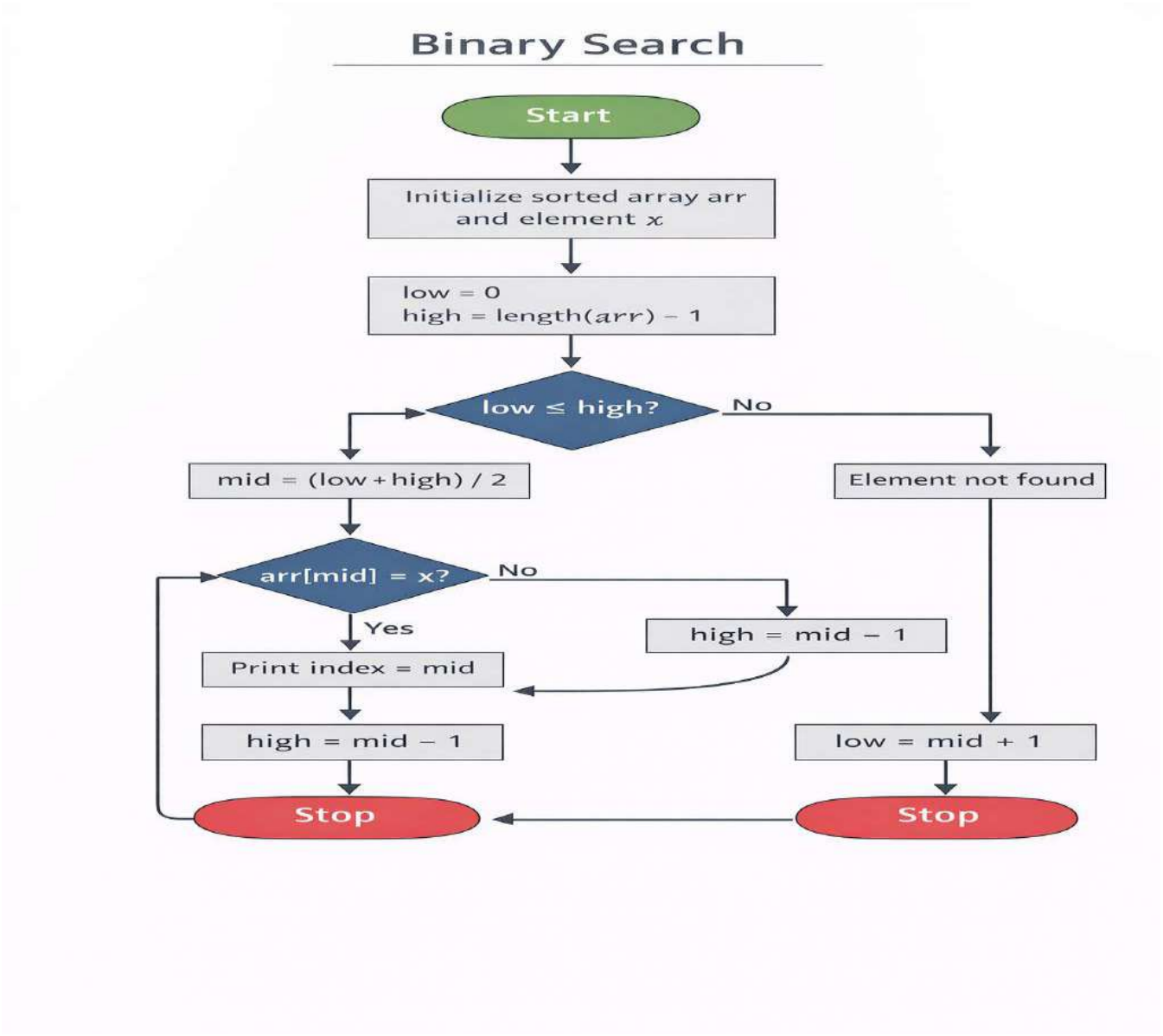
```
def binary_search(arr, low, high, x):

    if high >= low:
        mid = (high + low) // 2

        # If element is present at the middle
        if arr[mid] == x:
            return mid

        # If element is smaller than mid
        elif arr[mid] > x:
            return binary_search(arr, low, mid - 1, x)

        # Else element can only be in right subarray
        else:
            return binary_search(arr, mid + 1, high, x)

    else:
        # Element is not present
        return -1
# Test array
arr = [2, 3, 4, 10, 40]
x = 10
# Function call
result = binary_search(arr, 0, len(arr) - 1, x)
```

```
if result != -1:
    print("Element is present at index", result)
else:
    print("Element is not present in array")
```

Program Output :

```
Element is present at index 3
```

Flowchart :



EXPERIMENT NO.7

AIM

To write a Python Program to perform selection sort.

Algorithm (Selection Sort)

1. Start
2. Create a list `alist` with unsorted elements.
3. Set `i` from last index to first index.
4. Initialize `pos = 0`.
5. Compare elements from index `1` to `i`.
6. If `alist[location] > alist[pos]`, update `pos = location`.
7. Swap `alist[i]` with `alist[pos]`.
8. Repeat steps 3–7 until the list is sorted.
9. Print the sorted list.
10. Stop.

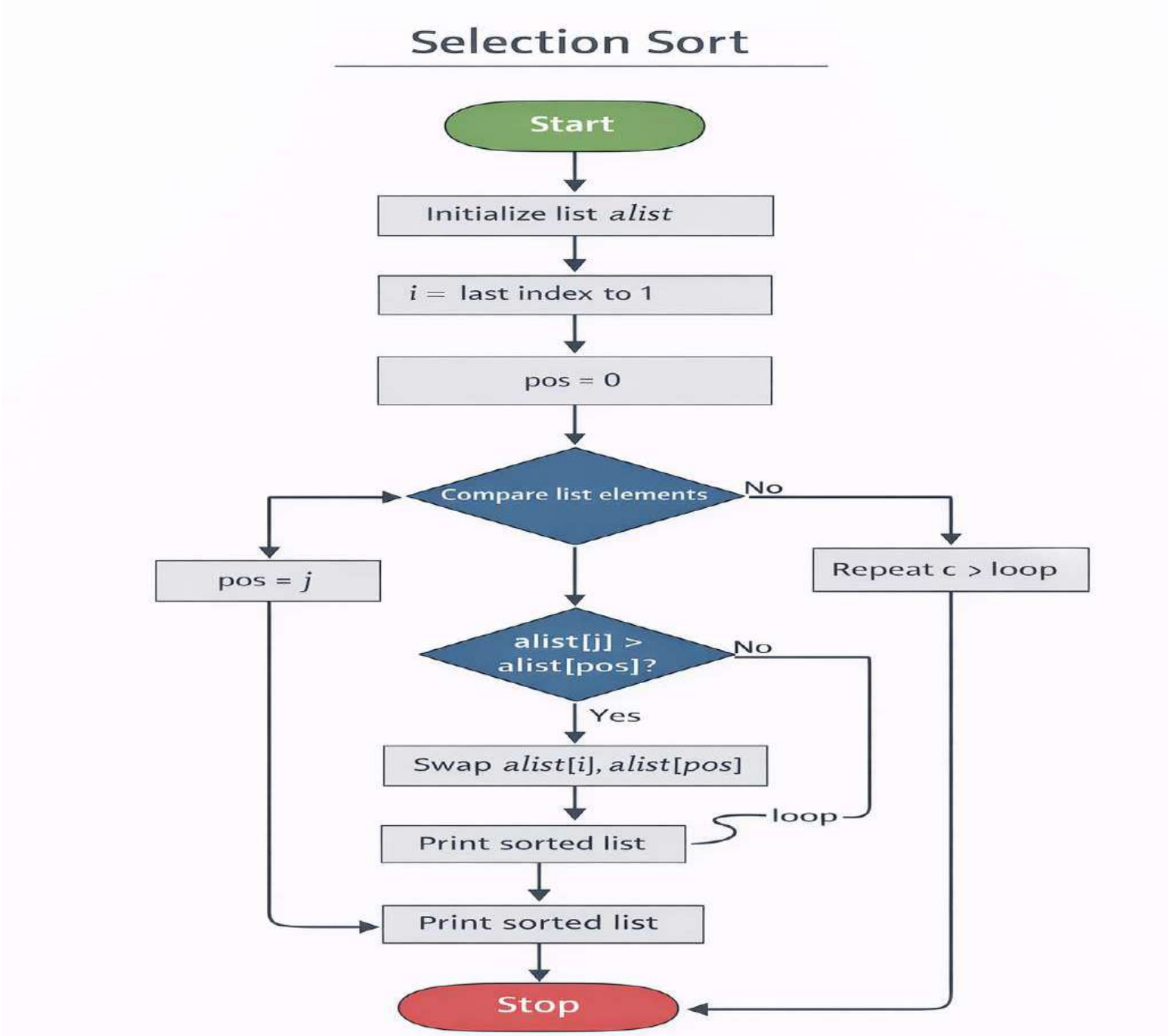
Program :

```
 def selectionSort(alist):

    for i in range(len(alist) - 1, 0, -1):
        pos = 0

        for location in range(1, i + 1):
            if alist[location] > alist[pos]:
                pos = location

        temp = alist[i]
        alist[i] = alist[pos]
        alist[pos] = temp


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

selectionSort(alist)

print(alist)
```

Sample Output:

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

Flowchart (Selection Sort) :



EXPERIMENT NO.8

AIM:

To write a Python Program to perform insertion sort.

Algorithm (Insertion Sort):

1. Start
2. Create a function named `insertionSort`.
3. Traverse the list from index `1` to `len(alist) - 1`.
4. Set `currentvalue = alist[index]`.
5. Set `position = index`.
6. While `position > 0` and `alist[position-1] > currentvalue`, do:
    - Move element one position ahead: `alist[position] = alist[position-1]`.
    - Decrease position: `position = position - 1`.
7. Insert `currentvalue` at correct position: `alist[position] = currentvalue`.
8. Repeat until the list is completely sorted.
9. Print the sorted list.
10. Stop.

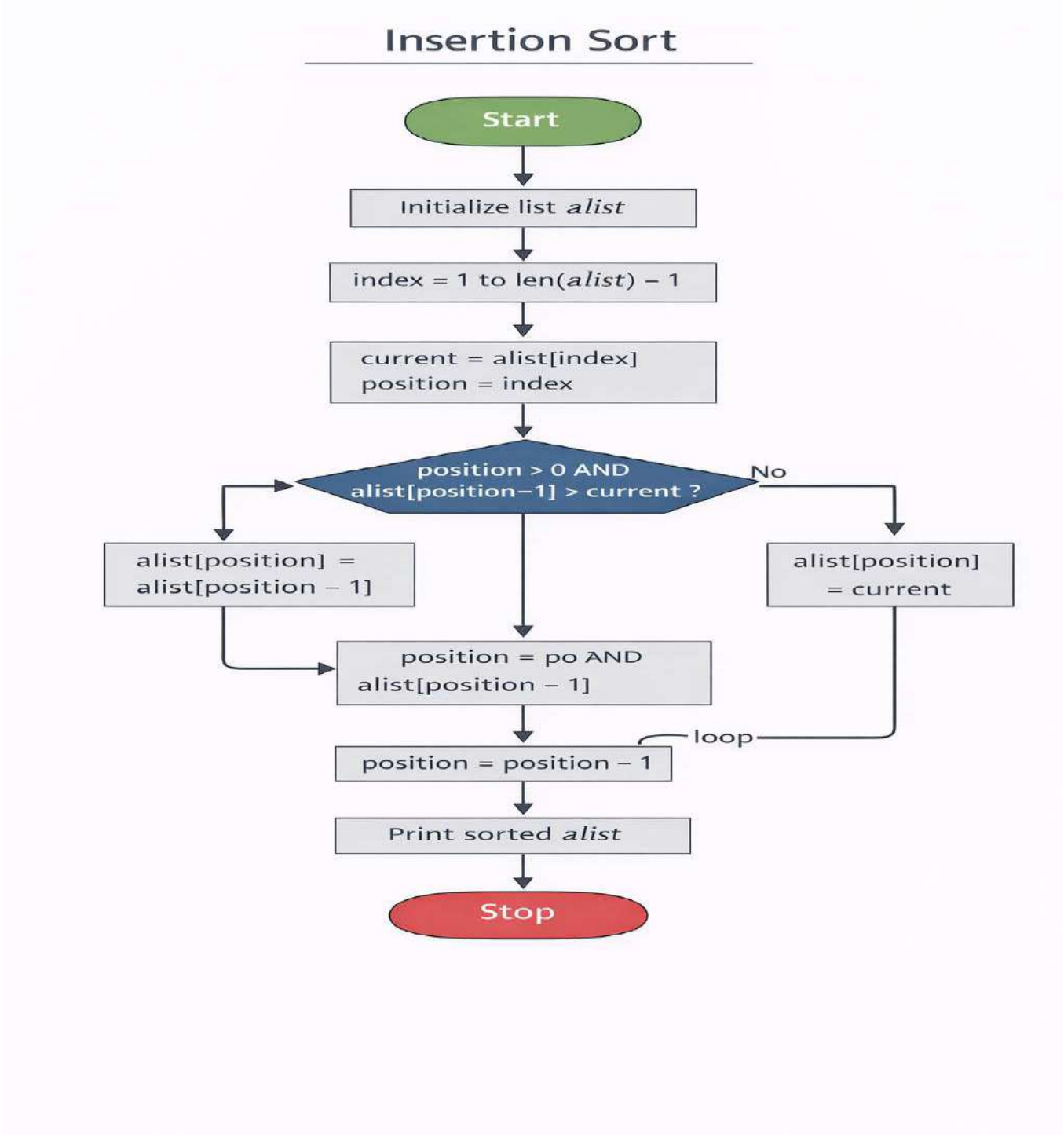
Program :

```
   def insertionSort(alist):

    for index in range(1, len(alist)):
        currentvalue = alist[index]
        position = index

        while position > 0 and alist[position - 1] > currentvalue:
            alist[position] = alist[position - 1]
            position = position - 1

        alist[position] = currentvalue


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

insertionSort(alist)

print(alist)
```

Sample Output:

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

Flowchart:


Insertion Sort
Start
Initialize list alist
index = 1 to len(alist) – 1
current = alist[index]
position = index
position > 0 AND
alist[position–1] > current ?
No
alist[position] =
alist[position – 1]
alist[position]
= current
position = po AND
alist[position – 1]
loop
position = position – 1
Print sorted alist
Stop

EXPERIMENT NO. 9

AIM :

To write a Python program to perform Merge Sort.

Algorithm (Merge Sort):

1. Start
2. Create a function `mergeSort(arr, l, r)`.
3. If `l < r`, find the middle index
    - `m = (l + r) // 2`
4. Recursively divide the array into two halves:
    - `mergeSort(arr, l, m)`
    - `mergeSort(arr, m+1, r)`
5. Merge the two sorted halves using `merge(arr, l, m, r)`.
6. In `merge()`:
    - Create two temporary arrays `L` and `R`.
    - Copy elements into these arrays.
    - Compare elements from both arrays.
    - Insert the smaller element into the original array.
7. Copy remaining elements if any.
8. Print the sorted array.
9. Stop.

Program :

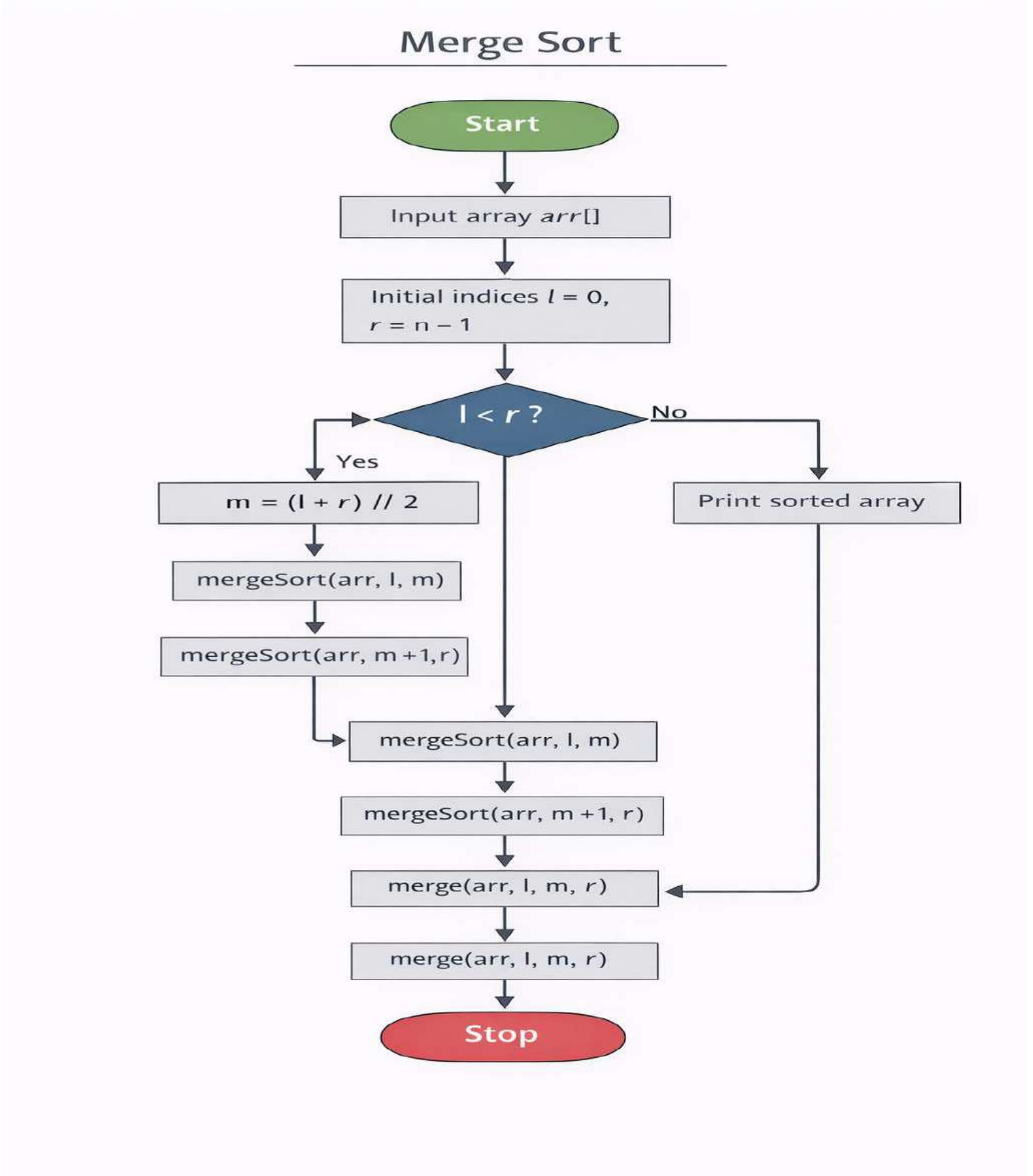
```
# Merge function
def merge(arr, l, m, r):

    n1 = m - l + 1
    n2 = r - m

    L = [0] * n1
    R = [0] * n2

    for i in range(0, n1):
        L[i] = arr[l + i]

    for j in range(0, n2):
        R[j] = arr[m + 1 + j]

    i = 0
    j = 0
    k = l

    while i < n1 and j < n2:
        if L[i] <= R[j]:
            arr[k] = L[i]
            i += 1
```

```
        else:
            arr[k] = R[j]
            j += 1
        k += 1

    while i < n1:
        arr[k] = L[i]
        i += 1
        k += 1

    while j < n2:
        arr[k] = R[j]
        j += 1
        k += 1


# Merge Sort function
def mergeSort(arr, l, r):

    if l < r:

        m = (l + r) // 2

        mergeSort(arr, l, m)
        mergeSort(arr, m + 1, r)
        merge(arr, l, m, r)


# Main program
arr = [2, 5, 3, 8, 6, 5, 4, 7]

n = len(arr)

mergeSort(arr, 0, n - 1)

print("Sorted array is:")

for i in range(n):
    print(arr[i], end=" ")
```

Sample output :

```
Sorted array is:
2 3 4 5 5 6 7 8
```

Flowchart :


Merge Sort
Start
Input array arr[]
Initial indices l = 0,
r = n − 1
l < r ?
No
Yes
m = (l + r) // 2
Print sorted array
mergeSort(arr, l, m)
mergeSort(arr, m +1,r)
mergeSort(arr, l, m)
mergeSort(arr, m +1, r)
merge(arr, l, m, r)
merge(arr, l, m, r)
Stop

EXPERIMENT NO.10

AIM :

To write a Python program to find first n prime numbers.

Algorithm :

1. Start
2. Read the value of n.
3. Print the message **"Prime numbers are"**.
4. For num in range 0 to n, perform the following:
5. If num > 1, check divisibility.
6. For i in range 2 to num-1:
    - If num % i == 0, then break the loop.
7. If the loop completes without break, print the value of num.
8. Repeat the process until num = n.
9. Stop.

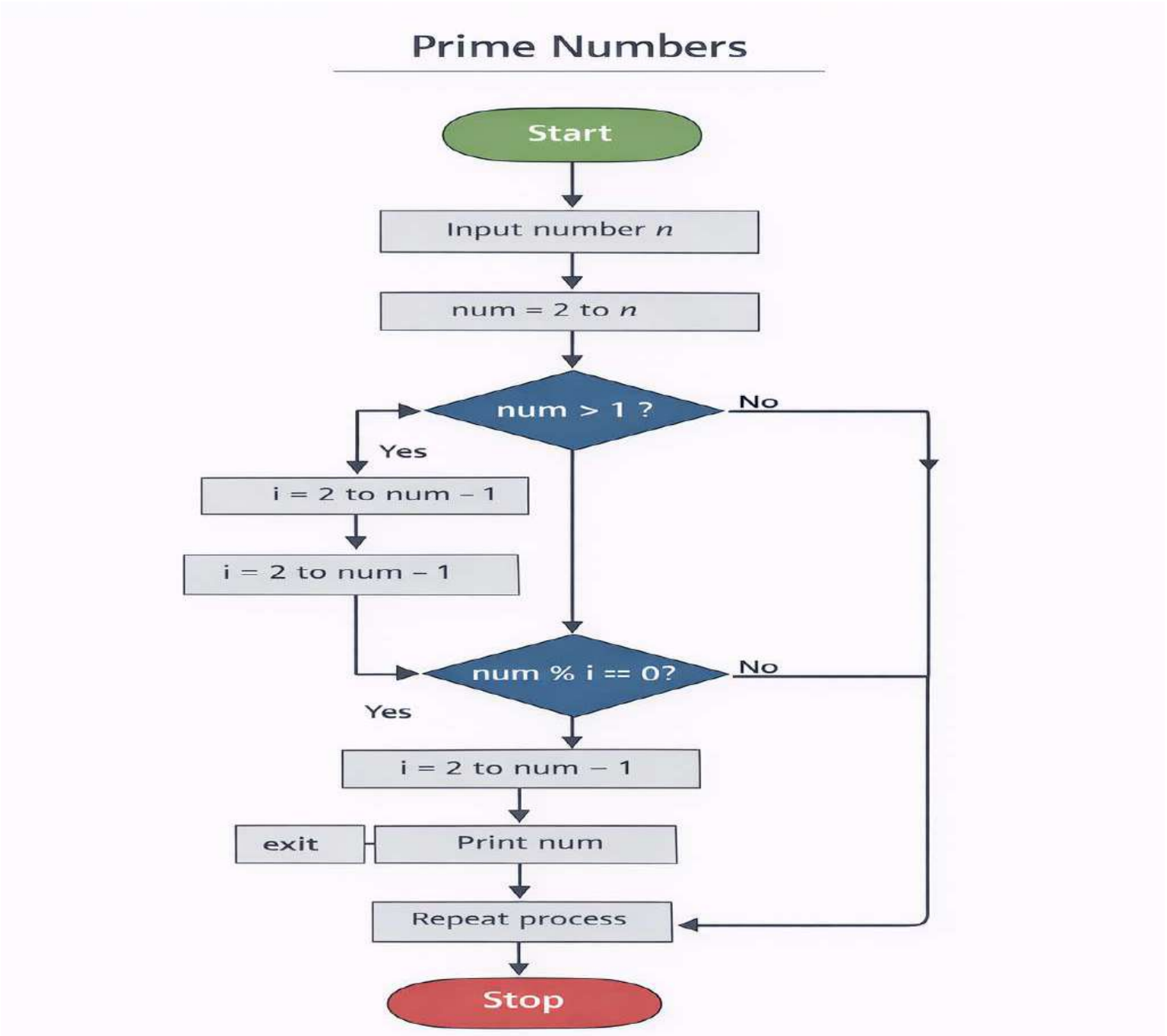
Program :

```
n = int(input("Enter the upper limit: "))

print("Prime numbers are")

for num in range(0, n + 1):

    if num > 1:

        for i in range(2, num):
            if num % i == 0:
                break
        else:
            print(num, end=" ")
```

Sample Output:

```
Enter the upper limit: 20
Prime numbers are
2 3 5 7 11 13 17 19
```

Flowchart :


Prime Numbers
Start
Input number n
num = 2 to n
num > 1 ?
No
Yes
i = 2 to num – 1
i = 2 to num – 1
num % i == 0?
No
Yes
i = 2 to num – 1
exit
Print num
Repeat process
Stop

EXPERIMENT NO.11

AIM:

To write a Python program to multiply matrices.

Algorithm :

1. Start
2. Input the size `n` of the matrix (for an `n × n` matrix).
3. Input elements of first matrix **A** using nested loops.
4. Input elements of second matrix **B** using nested loops.
5. Create a result matrix **result** initialized with `0`.
6. Use three nested loops:
    - Outer loop for rows of matrix **A**
    - Middle loop for columns of matrix **B**
    - Inner loop for multiplication and addition
7. Multiply elements `A[i][k] * B[k][j]` and add to `result[i][j]`.
8. Display the resultant matrix.
9. Stop.

Program :

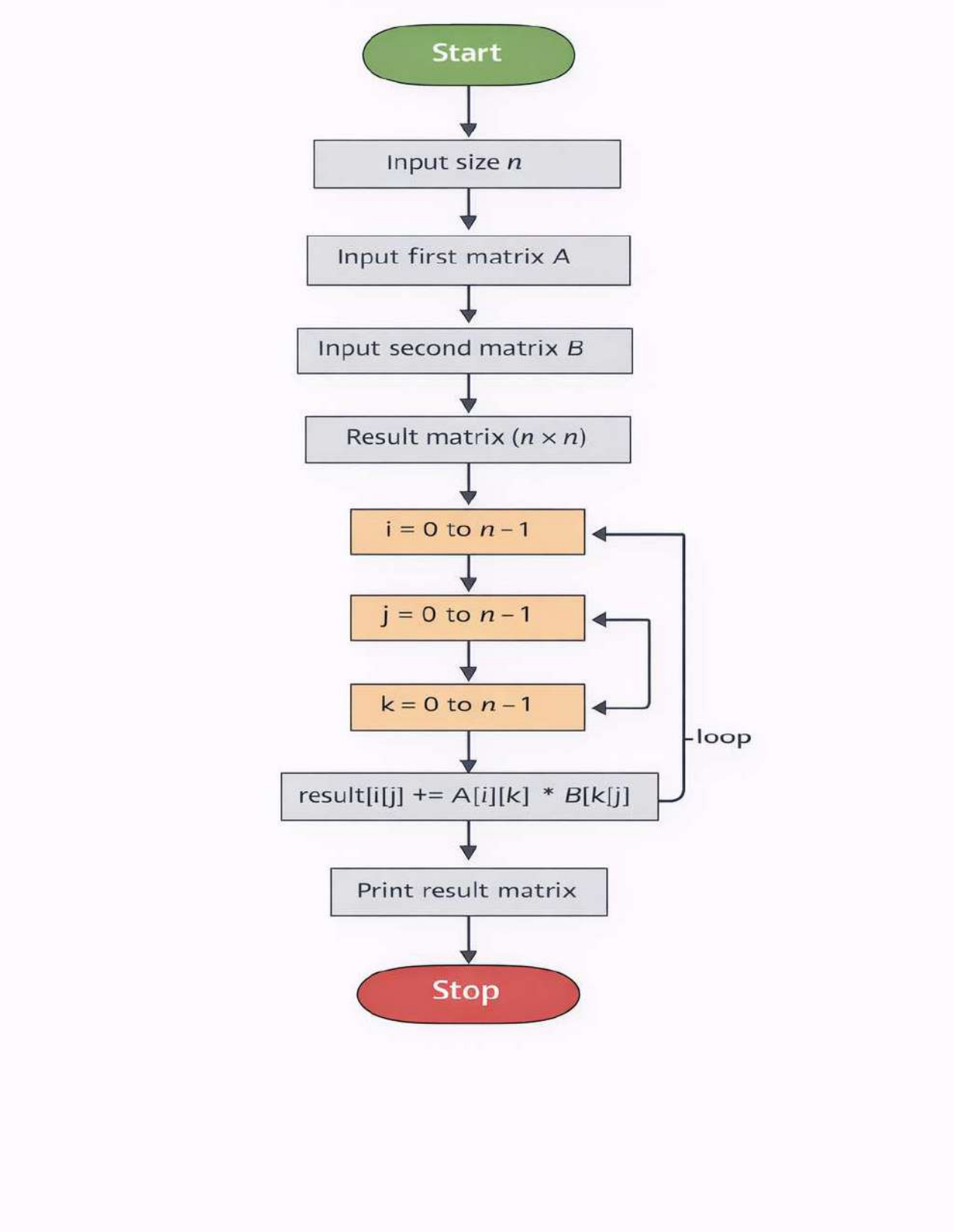
```
A = []
n = int(input("Enter N for N x N matrix: "))

print("Enter elements of first matrix:")

for i in range(n):
    row = []
    for j in range(n):
        row.append(int(input()))
    A.append(row)

print("First Matrix:")
for i in range(n):
    for j in range(n):
        print(A[i][j], end=" ")
    print()


B = []

print("Enter elements of second matrix:")

for i in range(n):
    row = []
    for j in range(n):
```

```
        row.append(int(input()))
    B.append(row)

print("Second Matrix:")
for i in range(n):
    for j in range(n):
        print(B[i][j], end=" ")
    print()


result = [[0 for j in range(n)] for i in range(n)]

for i in range(len(A)):
    for j in range(len(B[0])):
        for k in range(len(B)):
            result[i][j] += A[i][k] * B[k][j]


print("The Resultant Matrix Is:")

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

Sample Output:

```
Enter N for N x N matrix: 2

Enter elements of first matrix:
1
2
3
4

First Matrix:
1 2
3 4

Enter elements of second matrix:
5
6
7
8

Second Matrix:
5 6
7 8

The Resultant Matrix Is:
[19, 22]
[43, 50]
```

Flowchart :


Start
Input size n
Input first matrix A
Input second matrix B
Result matrix (n × n)
i = 0 to n – 1
j = 0 to n – 1
k = 0 to n – 1
result[i[j] += A[i][k] * B[k[j]
loop
Print result matrix
Stop

EXPERIMENT NO :12

Aim :

To write a Python program for command line arguments.

Algorithm:

1. Start
2. Import required modules sys and getopt.
3. Define a main() function.
4. Initialize variables inputfile and outputfile.
5. Use getopt.getopt() to read command line arguments.
6. If -h option is used, display help message and exit.
7. If -i or --ifile option is used, store input file name.
8. If -o or --ofile option is used, store output file name.
9. Print the input and output file names.
10. Call main() using sys.argv.
11. Stop.

Program:

```
#!/usr/bin/python3

import sys
import getopt

def main(argv):

    inputfile = ''
    outputfile = ''

    try:
        opts, args = getopt.getopt(argv, "hi:o:", ["ifile=", "ofile="])

    except getopt.GetoptError:
        print("test.py -i <inputfile> -o <outputfile>")
        sys.exit(2)

    for opt, arg in opts:

        if opt == '-h':
            print("test.py -i <inputfile> -o <outputfile>")
            sys.exit()

        elif opt in ("-i", "--ifile"):
```

```
            inputfile = arg

        elif opt in ("-o", "--ofile"):
            outputfile = arg

    print("Input file is:", inputfile)
    print("Output file is:", outputfile)


if __name__ == "__main__":
    main(sys.argv[1:])
```

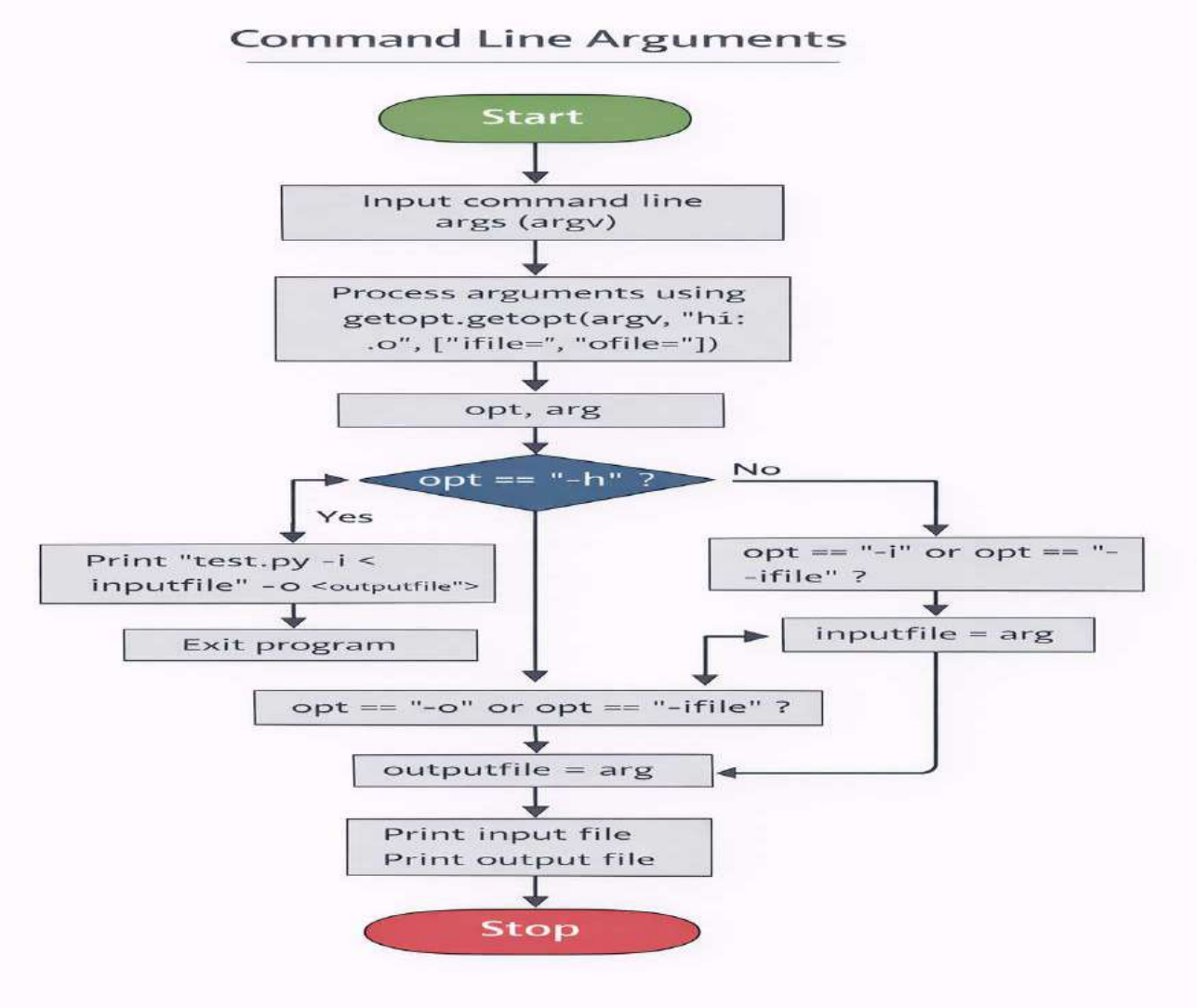
Example Command (Terminal)

```
    python test.py -i input.txt -o output.txt
```

Sample Output:

```
Input file is: input.txt
Output file is: output.txt
```

Flowchart:



EXPERIMENT NO.13

AIM:

To write a Python program to find the most frequent words in a text read from a file.

Algorithm:

1. Variable max Count will store the count of most repeated word.
2. Open a file in read mode using file pointer.
3. Read a line from file. Convert each line into lowercase and remove the punctuation marks.
4. Split the line into words and store it in an array.
5. Use two loops to iterate through the array. Outer loop will select a word which needs to be count. Inner loop will match the selected word with rest of the array. If match found, increment count by 1.
6. If count is greater than maxCount then ,store value of count in maxCount and corresponding word in variable word.
7. At the end, maxCount will hold the maximum count and variable word will hold most repeated word.

Program:

```
count = 0
word = ""
maxCount = 0
words = []

# Open file in read mode
file = open("data.txt", "r")

# Read each line from file
for line in file:

    # Convert to lowercase and remove punctuation
    string = line.lower().replace(',', '').replace('.', '').split()

    # Add words to list
    for s in string:
        words.append(s)

# Find most repeated word
for i in range(0, len(words)):
    count = 1

    for j in range(i + 1, len(words)):
        if words[i] == words[j]:
            count = count + 1
```

```
    if count > maxCount:
        maxCount = count
        word = words[i]

print("Most repeated word:", word)

file.close()
```

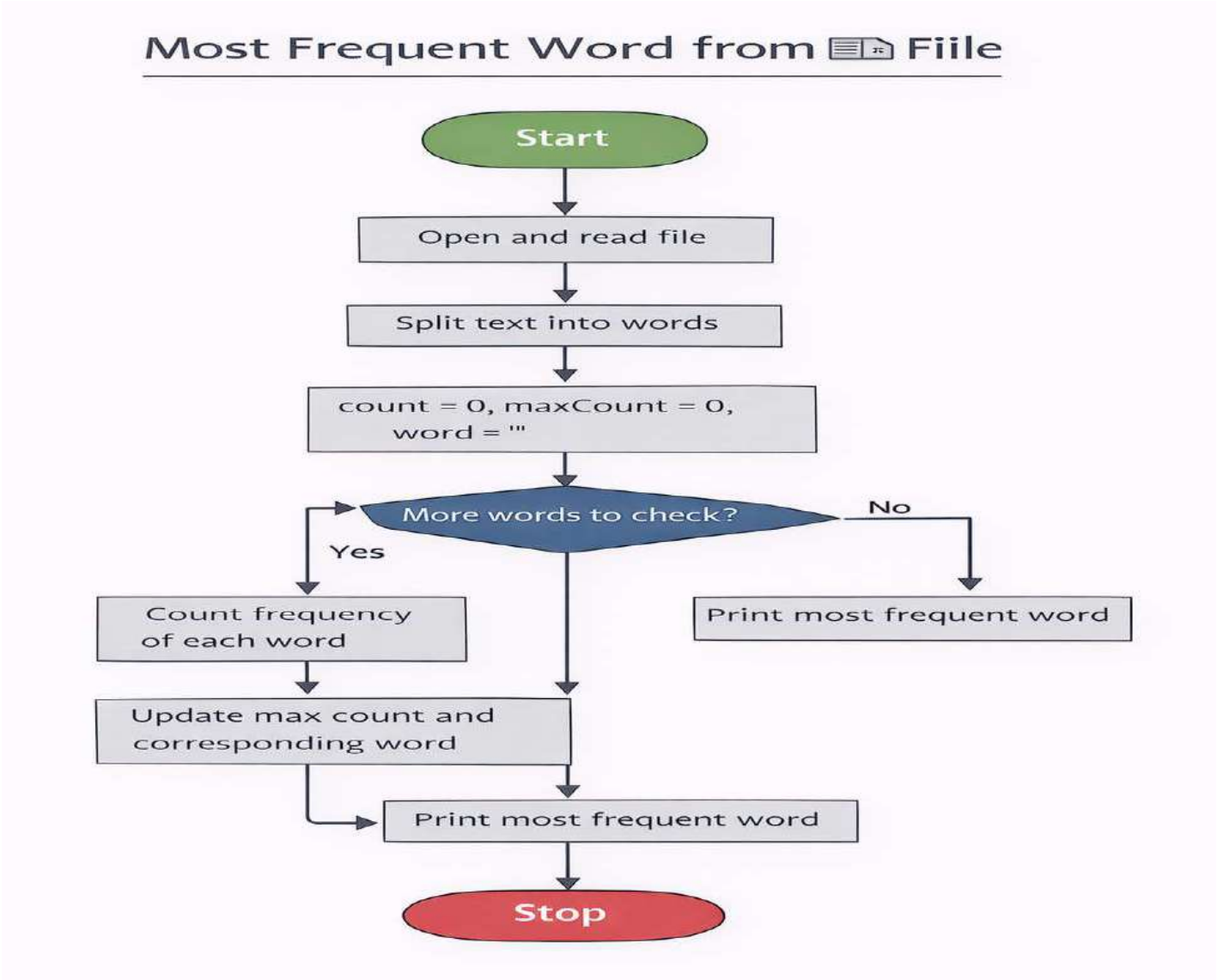
Example File (data.txt) :

```
Computer is a machine.
Computer can perform many tasks.
Computer is very useful.
```

Sample Output:

```
    Most repeated word: computer
```

Flowchart:



EXPERIMENT NO.14

Aim:

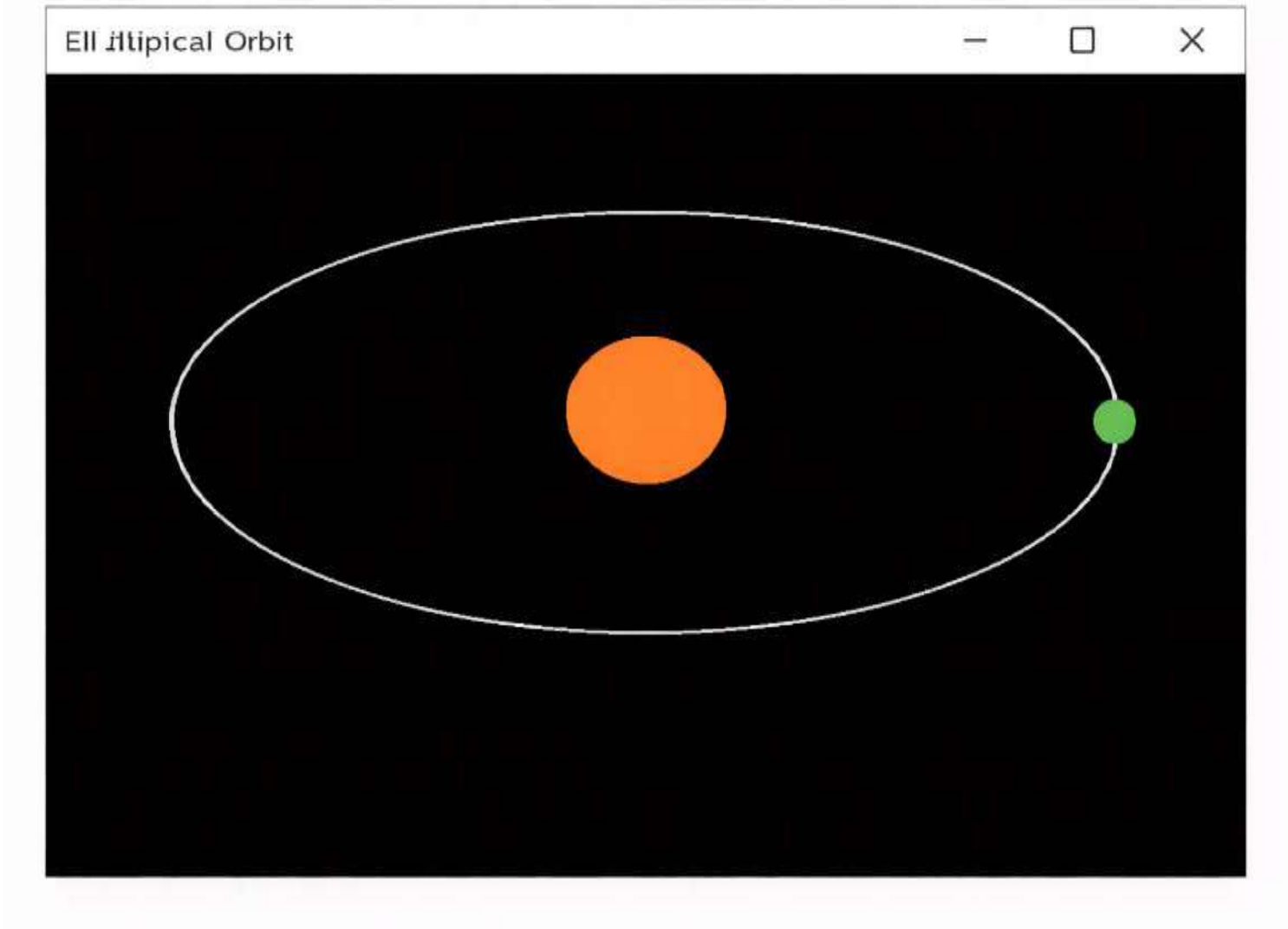
To write a Python program to simulate elliptical orbits using Pygame.

Program:

```
import pygame
import math
import sys

pygame.init()

# Setting screen size
screen = pygame.display.set_mode((600, 300))

# Setting caption
pygame.display.set_caption("Elliptical Orbit")

# Creating clock variable
clock = pygame.time.Clock()

xRadius = 250
yRadius = 100

degree = 0

while True:

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            sys.exit()

    # Calculate position on ellipse
    x1 = int(math.cos(degree * 2 * math.pi / 360) * xRadius) + 300
    y1 = int(math.sin(degree * 2 * math.pi / 360) * yRadius) + 150

    screen.fill((0, 0, 0))

    # Draw sun
    pygame.draw.circle(screen, (255, 69, 0), (300, 150), 40)

    # Draw elliptical orbit
    pygame.draw.ellipse(screen, (255, 255, 255), (50, 50, 500, 200), 1)

    # Draw moving planet
    pygame.draw.circle(screen, (0, 255, 0), (x1, y1), 20)

    pygame.display.flip()
```

```
    degree = (degree + 10) % 360

    clock.tick(5)
```

Output:



EXPERIMENTNO.:15

AIM:

To write a Python program to simulate a bouncing ball using Pygame.

Program:

```
import os
os.environ['PYGAME_HIDE_SUPPORT_PROMPT'] = "hide"

import sys
import pygame
from pygame.locals import *

pygame.init()

# Ball speed
speed = [1, 1]

# Background color
color = (255, 250, 250)

# Window size
width = 550
height = 300

# Create screen
screen = pygame.display.set_mode((width, height))
pygame.display.set_caption("Pygame Bouncing Ball")

# Load ball image
ball = pygame.image.load("ball.png")
rect_boundary = ball.get_rect()

while True:

    for event in pygame.event.get():
        if event.type == QUIT:
            pygame.quit()
            sys.exit()

    # Move ball
    rect_boundary = rect_boundary.move(speed)

    # Bounce from left and right wall
    if rect_boundary.left < 0 or rect_boundary.right > width:
        speed[0] = -speed[0]
```

```
    # Bounce from top and bottom wall
    if rect_boundary.top < 0 or rect_boundary.bottom > height:
        speed[1] = -speed[1]

    screen.fill(color)
    screen.blit(ball, rect_boundary)

    pygame.display.flip()
```

Sample Output: