Bharath

INSTITUTE OF HIGHER EDUCATION AND RESEARCH

(Declared as Deemed - to - be - University under section 3 of UGC Act 1956)

173, Agaram Road, Selaiyur, Chennai-600073

BHARATH INSTITUTE OF SCIENCE & TECHNOLOGY

Department of Computer Science & Engineering

PROBLEM SOLVING AND PHYTON

PROGRAMMING LABORATORY

PRACTICAL LAB RECORD

SUB CODE: U18ESCSIL1

Name : ERAKALA SAI TEJA

Year: I **Semester:** I **Batch: 2019-2023** **Section:** CSE-D

University Register No. U19CS301

35 YEARS OF EDUCATIONAL EXCELLENCE

NIRF RANKING NO.2 IN INDIA FOR OUTREACH & INCLUSIVITY. 2018

ACCREDITED BY NAAC WITH HIGHEST RATING A GRADE

APRIL '2019

Bharath

INSTITUTE OF HIGHER EDUCATION AND RESEARCH

(Declared as Deemed - to - be - University under section 3 of UGC Act 1956)

173, Agaram Road, Selaiyur, Chennai-600073

BHARATH INSTITUTE OF SCIENCE & TECHNOLOGY

Department of Computer Science & Engineering

PHYTON PROGRAMMING LABORATORY

PRACTICAL LAB RECORD

SUB CODE: U18ESCIL1

Name : ER AKALA SAI TEJA

Year: I **Semester:** I **Batch: 2019-2023** **Section:** CSE-D

University Register No. U19CS301

35 YEARS OF EDUCATIONAL EXCELLENCE

NIRF RANKING NO.2 IN INDIA

FOR OUTREACH & INCLUSIVITY. 2018

ACCREDITED BY **NAAC** WITH HIGHEST RATING A GRADE

APRIL '2019

Bharath

INSTITUTE OF HIGHER EDUCATION AND RESEARCH

(Declared as Deemed - to - be - University under section 3 of UGC Act 1956)

173, Agaram Road, Selaiyur, Chennai-600073

Department of Computer Science & Engineering

SUB CODE: U18ESCIL1

Name :

Year: **Semester:** **Batch: 2019-2023** **Section:**

University Register No. U19CS301

Certified that this is the bonafide record of work done by the above student in the **Phyton Programming Laboratory (Sub Code: BCS6L3)** during the month **APRIL'18**

Signature of Faculty-In-Charge

Signature of Head ofDepartment

Submitted for the practical Examinationheldon .....(Date ofExam)
at Bharath Institute of Higher Education & Research,Chennai-73.

Internal Examiner

External Examiner

INDEX

S.no	DATE	NAME OF THE EXPERIMENT	PAGE NO	MARKS	SIGN
1	28.8.19	Compute the GCD of two numbers			
2	29.8.19	Find the square root of a number (Newton's method)			
3	7.9.19	Exponentiation (power of a number)			
4	14.9.19	Find the maximum of a list of numbers			
5	21.9.19	Linear search and Binary search			
6	28.9.19	Selection sort, Insertion sort			
7	5.10.19	Merge sort			
8	12.10.19	First n prime numbers			
9	19.10.19	Multiply matrices			
10	4.11.19	Programs that take command line arguments(word count)			
11	4.11.19	Find the most frequent words in a text read from a file			
12	7.11.19	Simulate elliptical orbits in Pygame			
13	15.11.19	Simulate elliptical orbits in Pygame			

Ex no: 1
Date: 23.8.19

GCD OF TWO NUMBERS

Aim:

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

Algorithm:

1. Define a function namedcomputeGCD()
2. Find the smallest among the two inputs x and y
3. Perform the following step tillsmaller+1

   Check if ((x % i == 0) and (y % i == 0)), then assign GCD=i
4. Print the value of gcd

Program:

```
defcomputeGCD(x,y): if x
  >y:
    smaller = y
  else:
    smaller = x
  for i in range(1, smaller+1):
    if((x % i == 0) and (y % i == 0)):
      gcd = i

  return gcd

num1 =54
num2=24

# take input from the user
# num1 = int(input("Enter first number: "))
# num2 = int(input("Enter second number: "))

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

Sample output :

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

Result:

Thus the Python program to compute GCD of two numbers is executed successfully and the output is verified.

Ex no: 2
Date: 29.8.19

SQUARE ROOT OF A NUMBER

Aim:

To write a Python Program to find the square root of a number by Newton'sMethod.

Algorithm:

1. Define a function named newtonSqrt().
2. Initialize approx aso.5*nandbetter aso.5*(approx.+n/approx.)
3. Use a while loop with a condition better!=approx to perform the following,
   i. Set approx.=better
   ii. 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

print('The square root is' ,newtonSqrt(81))
```

Output:

The square root is 9

Result:

Thus the Python program for finding the square root of a given number by Newton's Method is executed successfully and the output is verified.

Ex no: 3
Date: 7.9.19

EXPONENTIATION OF A NUMBER

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)
    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: 7

Enter exponential value: 2

Result:49

Result:

Thus the Python program to find the exponentiation of a number is executed successfully and the output is verified.

Ex no: 4 **FINDING MAXIMUM FROM A LIST OF NUMBERS**
Date: 14.9.19

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 i 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)
```

Output:

```
Enter the upperlimt3
Enter the numbers 6
Enter the numbers 9
Enter the numbers
90
The maximum number is 90
```

Result:

Thus a Python Program to find the maximum from the given list of numbers is successfully executed and the output is verified.

Ex no: 5(a) **LINEAR SEARCH**
Date: 21.9.19

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 of the item
4. Else increment the position and repeat step 3 until pos reaches the length of the list

Program:

```
def search(alist,item):
 pos=0
 found=False
 stop=False
 while pos<len(alist) and not found and not stop:
  if alist[pos]==item:
   found=True
   print("element found in position",pos)
  else:
   if alist[pos]>item:
    stop=True
   else:
    pos=pos+1
return found a=[]
n=int(input("enter upper limit")) for i
in range(0,n):
 e=int(input("enter the elements")) a.append(e)
x=int(input("enter element to search"))
search(a,x)
```

Output:

Enter upper limit 5

Enter the elements 6

Enter the elements 45

Enter the elements 2

Enter the elements 61

Enter the elements 26

Enter element to search

6

Element found in position

1

Result:

Thus the Python Program to perform linear search is executed successfully and the output is verified.

Ex no: 5(b) **BINARY SEARCH**
Date: 21.9.19

Aim:

To write a Python Program to perform binary search.

Algorithm:

1. Read the search element
2. Find the middle element in the sorted list
3. Compare the search element with the middle element
   i. if both are matching, print element found
   ii. else then check if the search element is smaller or larger than the middle element
4. If the search element is smaller than the middle element, then repeat steps 2 and 3 for the left sub list of the middle element
5. If the search element is larger than the middle element ,then repeat steps 2 and 3 for the right sub list of the middle element
6. Repeat the process until the search element if found in the list
7. If element is not found, loop terminates

Program:

```
    else:
        first=mid+mid-1
  return found


a=[]
n=int(input("enter upper limit"))
for i in range(0,n):
  e=int(input("enter the elements"))
  a.append(e)
x=int(input("enter element to search"))
bsearch(a,x)
```

Output:

```
enter upper limit 6
enter the elements 2
enter the elements 3
enter the elements 5
enter the elements 7
enter the elements 14
enter the elements 25
enter element to search 5
element found in position 2
```

Result:

Thus the Python Program to perform binary search is executed successfully and the output is verified

Ex no: 6
Date: 28.9.19

SELECTION SORT

Aim:

To write a Python Program to perform selection sort.

Algorithm:

1. Create a function named selection sort
2. Initialise pos=0
3. If a list[location]>a list[pos] then perform the following till i+1,
4. Set pos=location
5. Swap a list[i] and a list[pos]
6. Print the sorted list

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)
```

Output:

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

Result:

Thus the Python Program to perform selection sort is successfully executed and the output is verified.

Ex no: 6(b) **INSERTION SORT**

Date: 28.9.19

Aim:

To write a Python Program to perform insertion sort.

Algorithm:

1. Create a function named insertion sort
2. Initialise current value=a list[index] and position=index
3. While position>0anda list[position-1]>current value, perform the following till en(a list)
4. a list[position]=a list[position-1]
5. position =position-1
6. a list[position]=current value
7. Print the sorted list

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)
```

Output:

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

Result:

Thus the Python program to perform insertion sort is successfully executed and the output is verified.

Ex no: 7
Date: 5.10.19

MERGE SORT

Aim:

To write a Python Program to perform Merge sort.

Algorithm:

1. Create a function named merge sort
2. Find the mid of the list
3. Assign left half = a list[:mid] and right half =a list[mid:]
4. Initialise i=j=k=0
5. While i<len (left half)and j<len(right half),perform the following if
   left half[i] <right half[j]:
   a list[k]=left half[i]
   Increment i else
   a list[k]=right half[j]
   Increment j Increment k
6. While i< len(left half),perform the following a
   list[k]=left half[i]
   Increment i
   Increment k
7. While j<len(right half),perform the following a
   list[k]=right half[j]
   Increment j
   Increment k
8. Print the sorted list

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=j=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(a list)
```

Output:

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

Result:

Thus the Python program to perform merge sort is successfully executed and the output is verified

Ex no: 8
Date: 12-10-19

FIRST N PRIME NUMBERS

Aim:

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

Algorithm:

1. Read the value of n
2. for num in range(0,n + 1), perform the following
3. if num%i is 0 then break else
   print the value of num
4. Repeat step 3 for i inrange(2,num)

Program:

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

print("Prime numbers are")

for num in range(0,n + 1):
  #primenumbersaregreaterthan1 if
  num >1:
    for i in range(2,num): if
      (num % i) == 0:
        break
    else:
      print(num)
```

Output:

```
Enter the upper limit: 100
Prime numbers are

2
3
5
7
11
13
17
19
23
29
31
37
41
43
47
53
59
61
67
71
73
79
83
89
97
```

Result:

Thus the Python Program to find the first n prime numbers is executed successfully and the output is verified.

Ex no: 9
Date:

MATRIX MULTIPLICATIONS

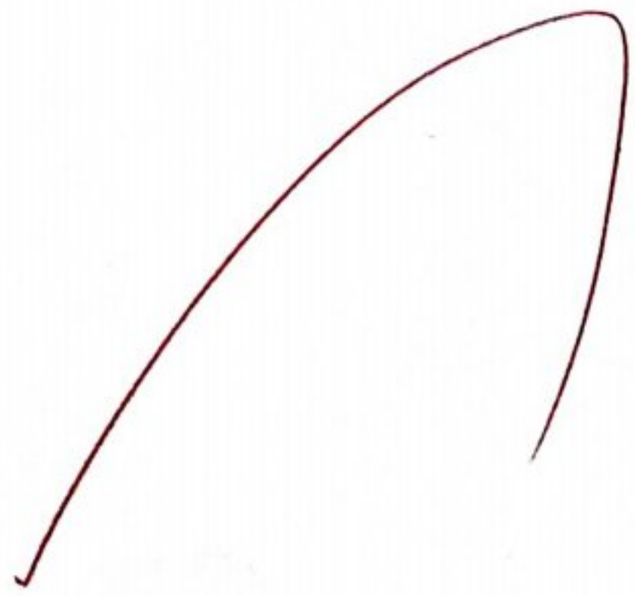
Aim:
To write a Python program to multiply matrices.

Algorithm:

1. Define two matrices X and Y
2. Create a resultant matrix named 'result'
3. for i in range(len(X)):
i. for j in range(len(Y[o])):
a) for k in range(len(Y))
b) result[i][j] += X[i][k] *Y[k][j]
4. for r in result, print the value of r

Program:

```
X = [[12,7,3],
[4 ,5,6],
[7 ,8,9]]
Y = [[5,8,1,2],
[6,7,3,0],
[4,5,9,1]]
result = [[0,0,0,0],
[0,0,0,0],
[0,0,0,0]]
for i in range(len(X)):
for j in range(len(Y[o])):
for k in range(len(Y)):
result[i][j] += X[i][k] * Y[k][j]
for r in result:
print(r)
```



Output:

```
[114, 160, 60, 27]

[74, 97, 73, 14]
```

[119, 157, 112, 23]

Result:

Thus the Python Program to find the matrix multiplication is executed successfully and the output is verified.

Output:

Enter upper limit 5

Enter the elements 6

Enter the elements 45

Enter the elements 2

Enter the elements 61

Enter the elements 26

Enter element to search

6

Element found in position

1

Result:

Thus the Python Program to perform linear search is executed successfully and the output is verified.

Ex no: 5(b) **BINARY SEARCH**

Date: 21.9.19

Aim:

To write a Python Program to perform binary search.

Algorithm:

1. Read the search element
2. Find the middle element in the sorted list
3. Compare the search element with the middle element
   i. if both are matching, print element found
   ii. else then check if the search element is smaller or larger than the middle element
4. If the search element is smaller than the middle element, then repeat steps 2 and 3 for the left sub list of the middle element
5. If the search element is larger than the middle element ,then repeat steps 2 and 3 for the right sub list of the middle element
6. Repeat the process until the search element if found in the list
7. If element is not found, loop terminates

Program:

```
    else:
        first=mid+mid-1
 return found


a=[]
n=int(input("enter upper limit"))
for i in range(0,n):
 e=int(input("enter the elements"))
 a.append(e)
x=int(input("enter element to search"))
bsearch(a,x)
```

Ex no: 10

FINDING MOST FREQUENT WORDS IN A TEXT READ FROM A FILE

Date:

Aim:

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

Algorithm:

1. Read the file name
2. Open the file in read mode
3. Read each line from the file to count the lowers and words
4. Read each line from the file to replace the punctuations
5. Split each line into words and count them
6. Print the words and counts

Program:

```
def main():
   filename=raw_input("enter the file").strip()
   infile=open(filename,"r")
   wordcounts={} for
   line in infile:
         processLine(line.lower(),wordcounts)
   pairs=list(wordcounts.items())
   items=[[x,y] for (y,x) in pairs]
   items.sort()
   for i in range(len(items)-1,len(items)-11,-1):
      print(items[i][1]+"\t"+str(items[i][0]))
def processLine(line,wordcounts):
   line=replacePunctuations(line)
   words=line.split()
   for word in words:
      if word in wordcounts: wordcounts[word]+=1
      else:
         wordcounts[word]=1 def
replacePunctuations(line):
   for ch in line:
      if ch in "~@#$%^&*()_-+=<>?/.,:;!{}[]"":
         line=line.replace(ch," ")
   return line main()
```

Output:

```
Enter a filename:a.txt


Hi      1
How     1
Are     1
You     1
```

Result:

Thus the Python program to find the most frequent words in a text read from a file is executed successfully and the output is verified.

Ex no: 11

FINDING MOST FREQUENT WORDS IN A TEXT READ FROM A FILE

Date: 9.11.19

Aim:

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

Algorithm:

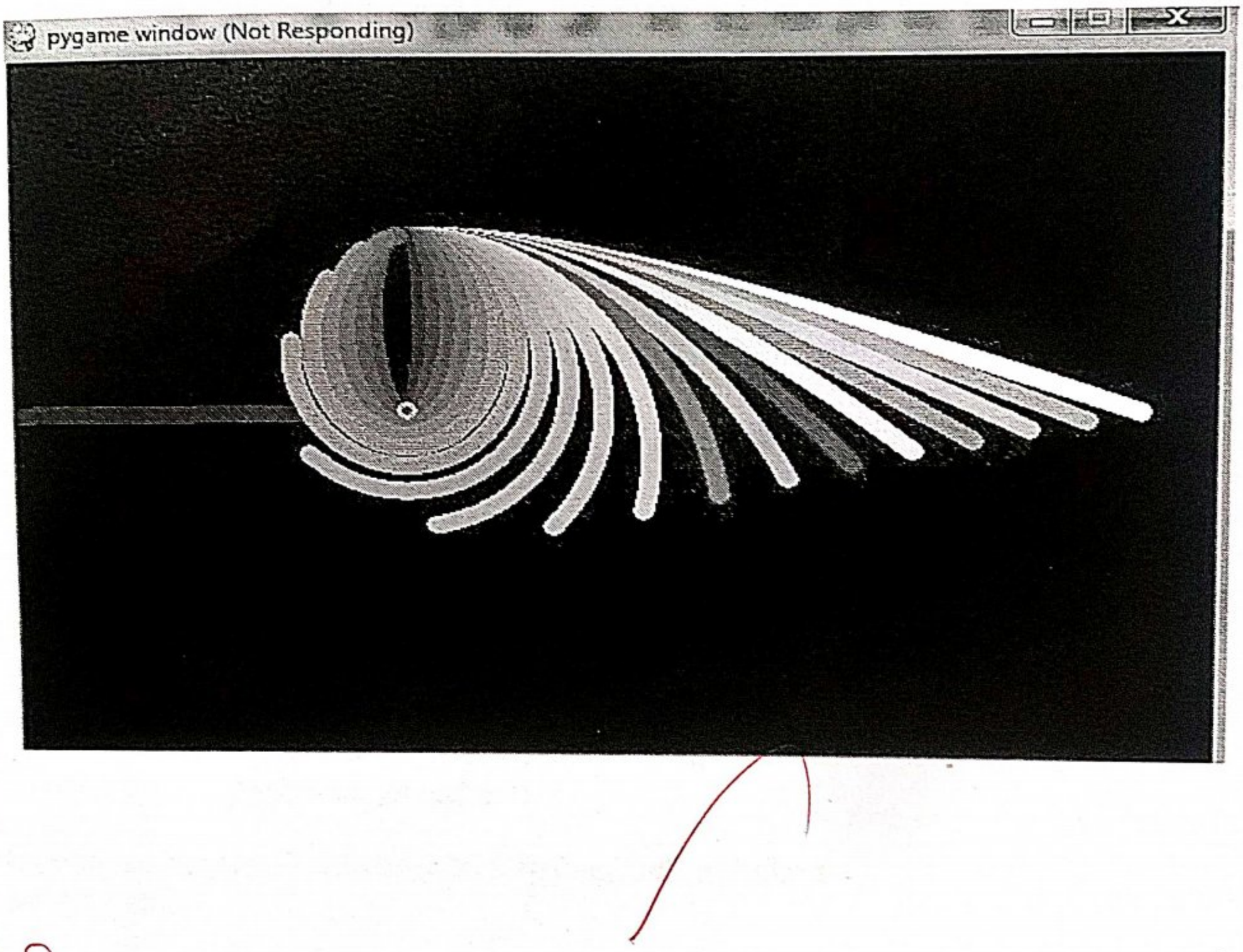
1. Import the required packages
2. Set up the colours for the elliptical orbits
3. Define the parameters to simulate elliptical orbits
4. Display the created orbits

Program:

```
import math
import random
import pygame
class Particle ():
    definit____(self,x,y,colour=0x000000):
        self.x =x
        self.y = y
        self.vx = 0
        self.vy = 0
        self.colour = colour
    def apply_gravity (self, target):
        dsqd=(self.x-target.x)**2+(self.y-target.y)**2
        #g=G*m/dsqd*normalized(self-target)
        if dsqd == 0:
            return
        self.vx+=-1/dsqd*(self.x-target.x)/dsqd**0.5
        self.vy+=-1/dsqd*(self.y-target.y)/dsqd**0.5
    def update (self):
        self.x += self.vx
        self.y += self.vy
pygame.init()
window = pygame.display.set_mode ((600, 400))
main_surface = pygame.Surface ((600, 400))
colours = [0x000000, 0x111111, 0x222222, 0x333333, 0x444444, 0x555555, 0x666666,
0x777777, 0x888888, 0x999999, 0xaaaaaa, 0xbbbbbb] + [0xFF0000, 0x00FF00, 0x0000FF,
```

```
    0xFFFF00, 0xFF00FF, 0x00FFFF, 0x888888, 0xFFFFFF, 0x808000, 0x008080, 0x800080,
    0x800000]
#colours = [0xFF0000, 0x00FF00, 0x0000FF, 0xFFFF00, 0xFF00FF, 0x00FFFF, 0x888888

    0xFFFFFF, 0x808000, 0x008080, 0x800080, 0x800000]
    particles = [Particle (200, 100, colours [i]) for i in range (20)]
    earth = Particle (200, 200)
    fori,pinenumerate(particles):
        p.vx = i /100
    while (True):
    #    main_surface.fill(0x000000)
        pygame.draw.circle (main_surface, 0x00FF00, (earth.x, earth.y), 5, 2)

    for p in particles:
        p.apply_gravity (earth)
        p.update ()
        pygame.draw.circle (main_surface, p.colour, (int (p.x), int (p.y)), 5, 2)
window.blit(main_surface, (0, 0))
pygame.display.flip()
```

Output:



Result:

Thus the Python Program to simulate elliptical orbits using Pygame is executed successfully and the output is verified

Ex no: 12

FINDING MOST FREQUENT WORDS IN A TEXT READ FROM A FILE

Date: 15.11.19

Aim:

To write a Python program to bouncing ball in Pygame.

Algorithm:

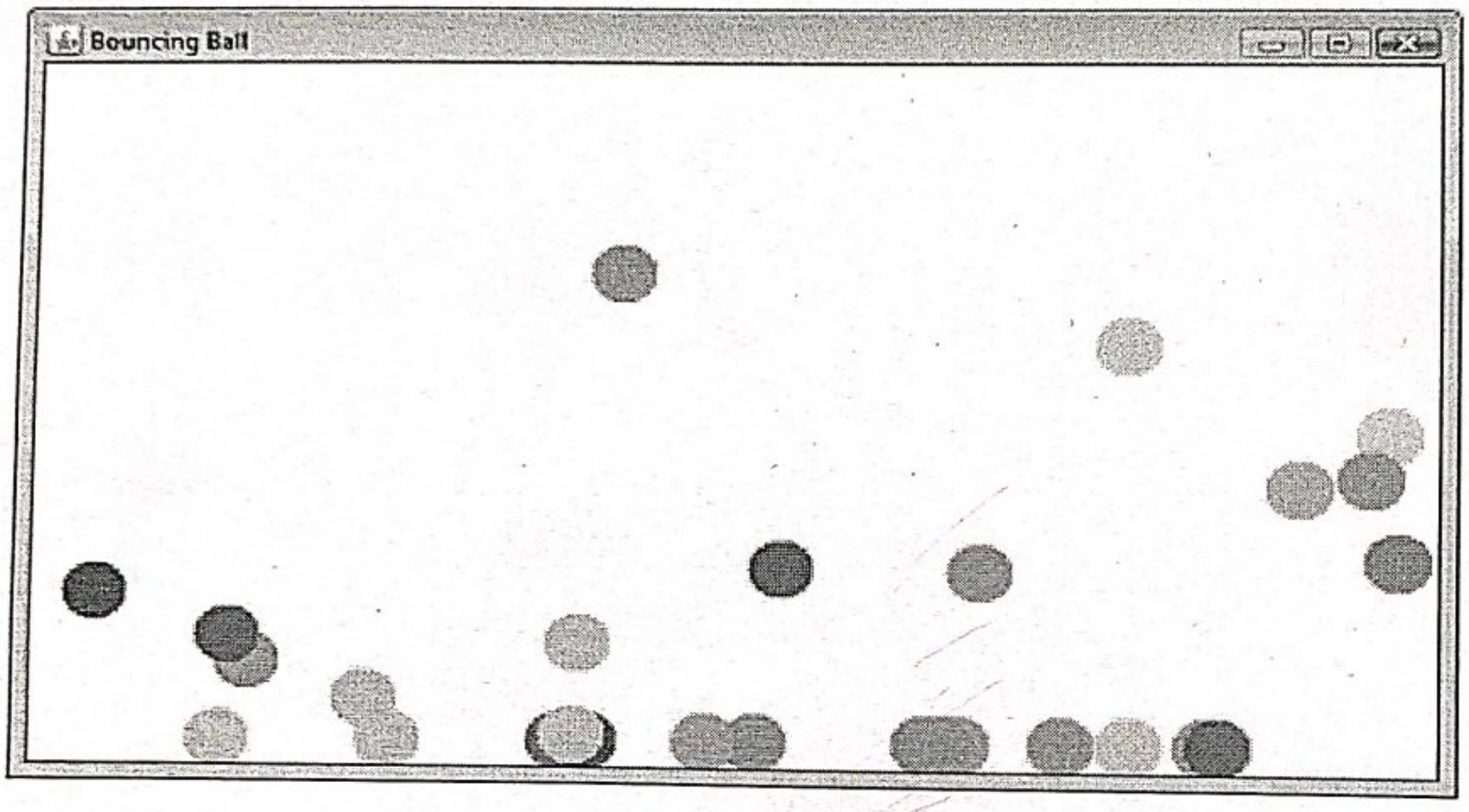
1. Import the required packages
2. Define the required variables
3. Define the screen space to display the bouncing balls in that space

Program:

```
import sys
import pygame
pygame.init()

size=width,height=320,240
speed = [2,2]
black=0,0,0

screen = pygame.display.set_mode(size)

ball = pygame.image.load('C:\\Users\\admin\\Desktop//ball.jpg') ballrect =
ball.get_rect()

while 1:
   for event in pygame.event.get():
      if event.type == pygame.QUIT: sys.exit()

   ballrect = ballrect.move(speed)
   ifballrect.left<oorballrect.right>width:
      speed[o] =-speed[o]
```

```
ifballrect.top<0orballrect.bottom>height:
  speed[1] =-speed[1]

screen.fill(black)
screen.blit(ball, ballrect)
pygame.display.flip()
```

Output:



Result:

Thus the Python Program to simulate bouncing ball using Pygame is executed successfully and the output is verified.