Sahakar Maharshi Bhausaheb Santuji Thorat

College Sangamner

DEPARTMENT OF COMPUTER SCIENCE

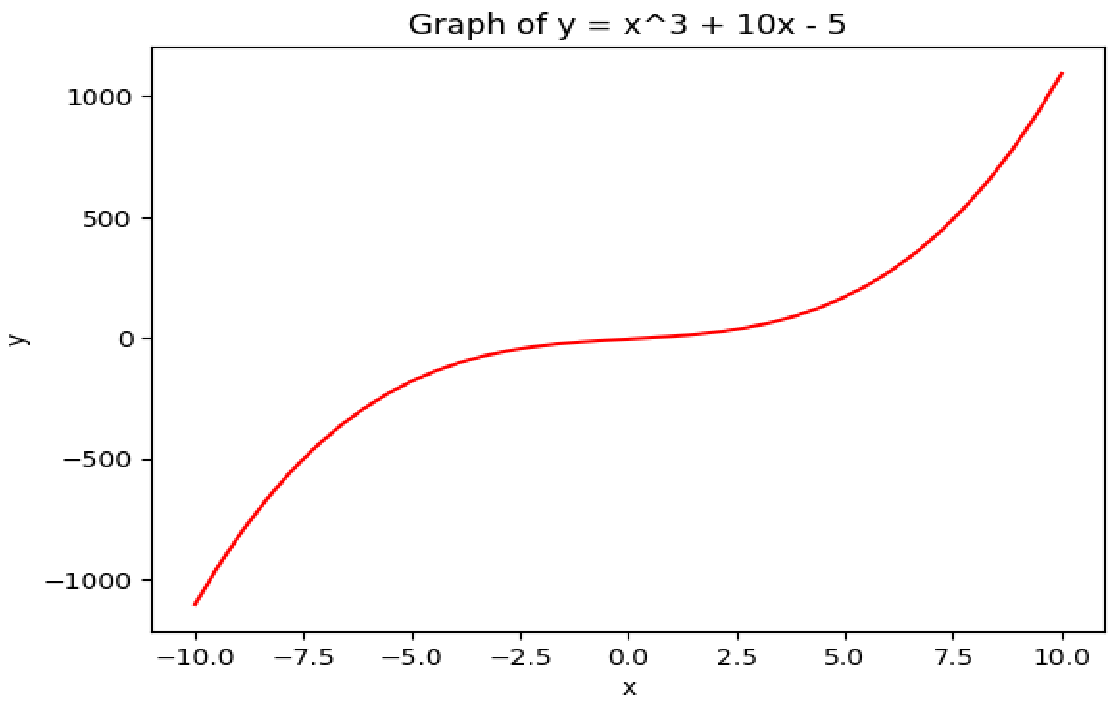
Sub : Mathematics

Remark
Demonstrator's
Signature
Date:- / /20

Name:-_Gorde Yash Somnath______ Roll.No:-__21_Date:-________
Title of the expt:-__Slip no 12______ Page.no:-_____Class:-_______BCS________

Q1. Attempt any Two of the following
A) Write a python program to plot the graph of $y=x^3+10x-5$ for x E[-10,10] in red color.
-→

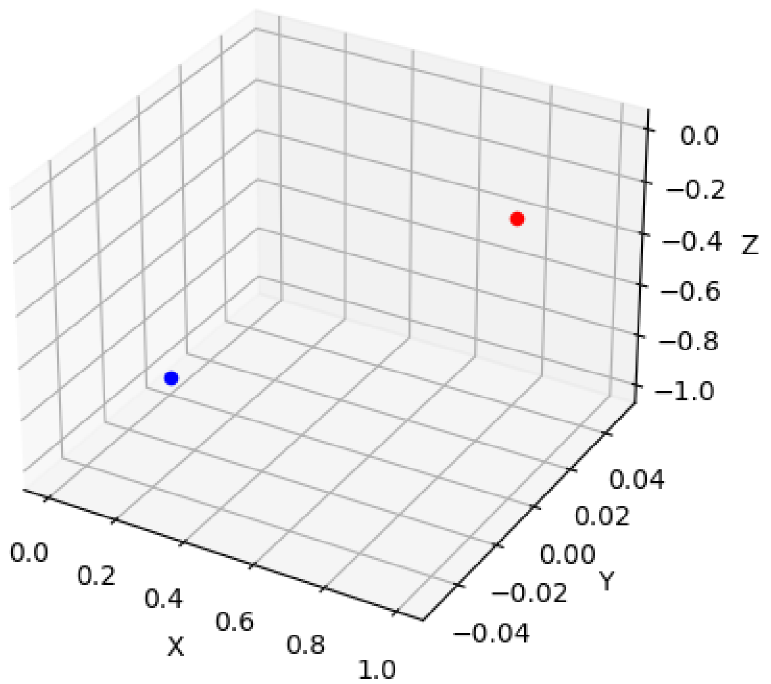
```
  import matplotlib.pyplot as plt
import numpy as np
def f(x):
   return x**3 + 10*x - 5

x_values = np.linspace(-10, 10, 1000)
y_values = f(x_values)
plt.plot(x_values, y_values, color='red')
plt.xlabel('x')
plt.ylabel('y')
plt.title('Graph of y = x^3 + 10x - 5')
plt.show()
```



B) Write a python program in 3D to rotate the point (1,0,0) through XZ –plane in clockwise direction (rotation through Y-axis by an angle of 90◦)

-→

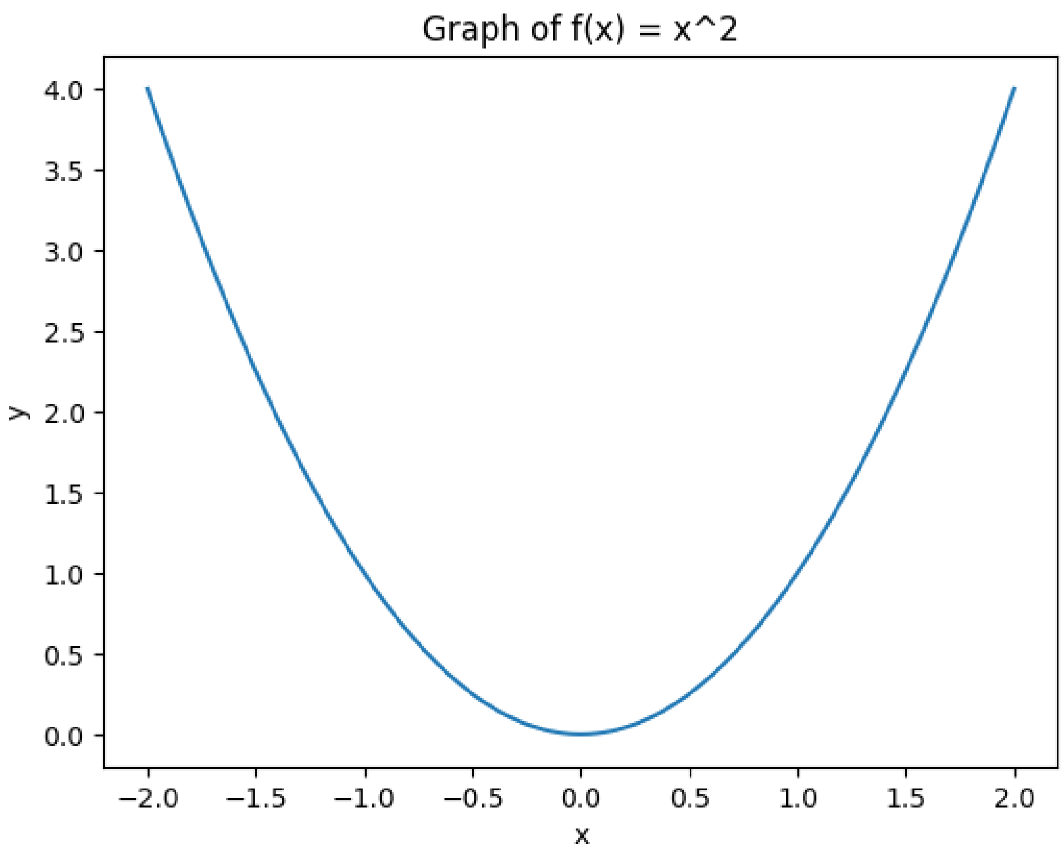
```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
point = np.array([[1], [0], [0]])
theta = np.radians(90)
cos_theta = np.cos(theta)
sin_theta = np.sin(theta)
rotation_matrix = np.array([[cos_theta, 0, sin_theta],
                            [0, 1, 0],
                            [-sin_theta, 0, cos_theta]])

rotated_point = rotation_matrix @ point
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.scatter(point[0], point[1], point[2], c='r', marker='o')
ax.scatter(rotated_point[0], rotated_point[1], rotated_point[2], c='b', marker='o')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
plt.show()
```



C) Using python plot the graph of function $f(x)=x^2$ on the interval [-2,2]

-→

```
import matplotlib.pyplot as plt
import numpy as np
def f(x):
    return x**2

x_values = np.linspace(-2, 2, 1000)
y_values = f(x_values)

plt.plot(x_values, y_values)
plt.xlabel('x')
plt.ylabel('y')
plt.title('Graph of f(x) = x^2')
plt.show()
```

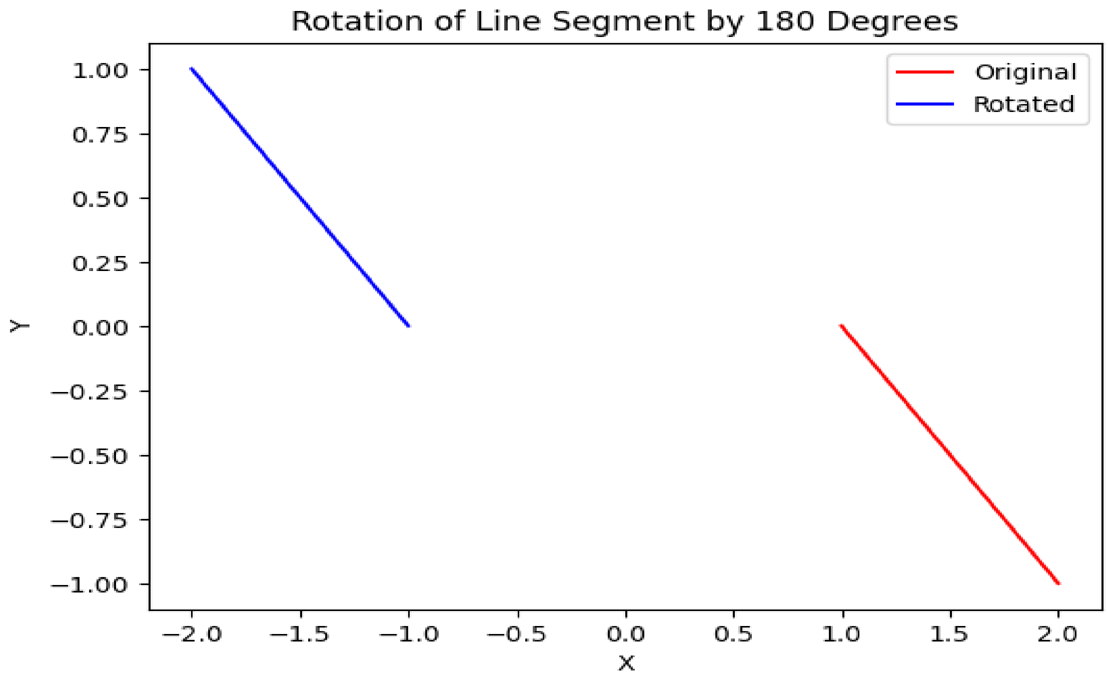


Q2) Attempt any TWO of the following

A) Write a python program to rotate the segment by 180◦ having end points (1,0) and (2,-1)

-→

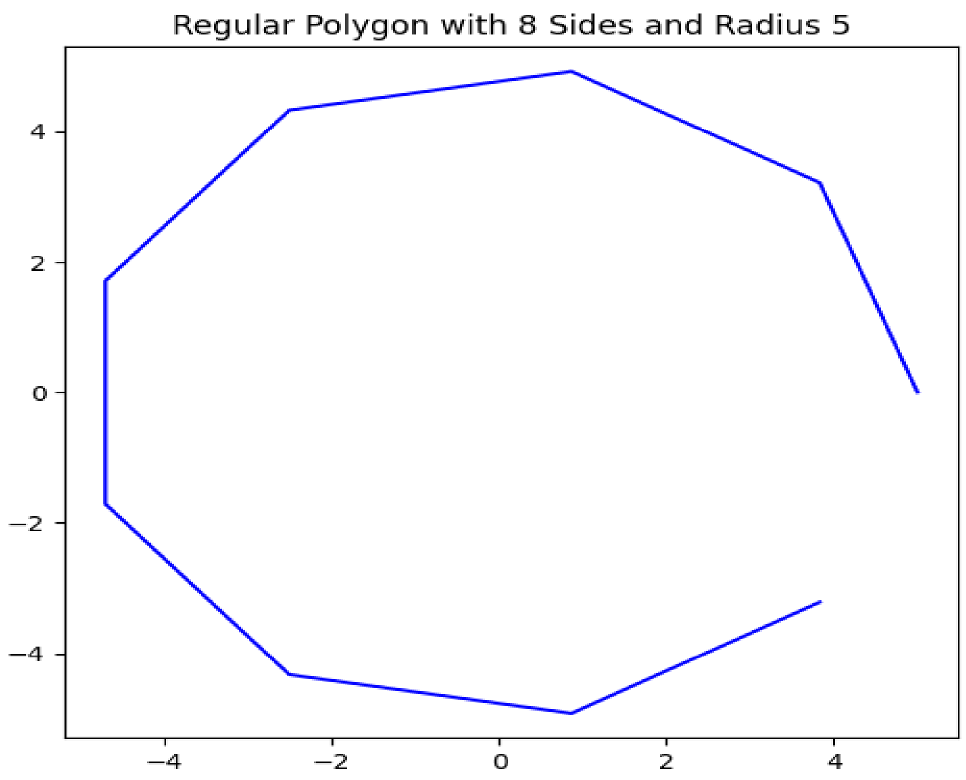
```
import numpy as np
import matplotlib.pyplot as plt
point1 = np.array([1, 0])
point2 = np.array([2, -1])

theta = np.radians(180)
cos_theta = np.cos(theta)
sin_theta = np.sin(theta)
rotation_matrix = np.array([[cos_theta, -sin_theta],
                            [sin_theta, cos_theta]])

rotated_point1 = rotation_matrix @ point1
rotated_point2 = rotation_matrix @ point2

plt.plot([point1[0], point2[0]], [point1[1], point2[1]], 'r', label='Original')
plt.plot([rotated_point1[0], rotated_point2[0]], [rotated_point1[1], rotated_point2[1]], 'b',
label='Rotated')
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Rotation of Line Segment by 180 Degrees')
plt.legend()
plt.show()
```



B) Write a python program to draw a polygon with 8 sides having radius 5 centered at origin and find its area and perimeter

-→

```
import matplotlib.pyplot as plt
import numpy as np

num_sides = 8
radius = 5

angles = np.linspace(0, 2 * np.pi, num_sides + 1)[:-1]

x_coords = radius * np.cos(angles)
y_coords = radius * np.sin(angles)

plt.figure(figsize=(6, 6))
plt.plot(x_coords, y_coords, 'b')
plt.axis('equal')
plt.title('Regular Polygon with 8 Sides and Radius 5')

side_length = 2 * radius * np.sin(np.pi / num_sides)
perimeter = num_sides * side_length
area = (num_sides * radius ** 2 * np.sin(2 * np.pi / num_sides)) / 2

print("Area:", area)
print("Perimeter:", perimeter)
plt.show()
```



C) Write a python program to find the area and perimeter of the ΔXYZ , where X(1,2), Y(2,-2),Z(-1,2)

-→

```
import math
X = (1, 2)
Y = (2, -2)
Z = (-1, 2)

XY = math.sqrt((Y[0] - X[0]) ** 2 + (Y[1] - X[1]) ** 2)
XZ = math.sqrt((Z[0] - X[0]) ** 2 + (Z[1] - X[1]) ** 2)
YZ = math.sqrt((Z[0] - Y[0]) ** 2 + (Z[1] - Y[1]) ** 2)

perimeter = XY + XZ + YZ
s = (XY + XZ + YZ) / 2
area = math.sqrt(s * (s - XY) * (s - XZ) * (s - YZ))

print("Area:", area)
print("Perimeter:", perimeter)
```

output :

```
Area: 4.000000000000003
Perimeter: 11.123105625617661
```

Q 3) Attempt the following

A) Attempt any ONE of the following

I) Write a program to solve the following LPP :

Min Z=3.5x+2y

Subject to x+y≥5

x≥4

y≤2

x,y≥0

-→

```
from scipy.optimize import linprog
c = [3.5, 2]
A = [[1, 1],
    [-1, 0],
    [0, 1]]
b = [5, -4, 2]

x_bounds = (0, None)
y_bounds = (0, None)
result = linprog(c, A_ub=A, b_ub=b, bounds=[x_bounds, y_bounds])
```

```
print("Optimal value:", round(result.fun, 2))
print("Optimal point:", tuple(round(x, 2) for x in result.x))
```

output :

```
Optimal value: 14.0
Optimal point: (4.0, 0.0)
```

II) Write a python program to solve the LPP and find optimal solution if exist

Max Z=3x+5y+4z

Subject to 2x+3y≤8

2y+5z≤10

3x+2y+4z≤15

X,y,z≥0

-→

```
from scipy.optimize import linprog
c = [-3, -5, -4]
A = [[2, 3, 0],
     [0, 2, 5],
     [3, 2, 4]]
b = [8, 10, 15]

x_bounds = (0, None)
y_bounds = (0, None)
z_bounds = (0, None)
result = linprog(c, A_ub=A, b_ub=b, bounds=[x_bounds, y_bounds, z_bounds])

if result.success:
    print("Optimal value:", round(-result.fun, 2))
    print("Optimal point:", tuple(round(x, 2) for x in result.x))
else:
    print("Optimization failed. The problem may be infeasible or unbounded.")
```

output :

```
Optimal value: 18.66
Optimal point: (2.17, 1.22, 1.51)
```

B) Attempt any ONE of the following

I) Write a python program to apply the following transformation on the point [-2,4]

A) Reflection through Reflection through –axis

B) Scaling in X-coordinate by factor 6

C) Scaling in Y-coordinate by factor 4.1

D) Shering in X-direction by $\frac{7}{2}$

-→

```
import numpy as np
point = np.array([-2, 4])
reflection_x = np.array([[1, 0], [0, -1]])
reflected_point_x = np.dot(reflection_x, point)

scaling_x = np.array([[6, 0], [0, 1]])
scaled_point_x = np.dot(scaling_x, point)
scaling_y = np.array([[1, 0], [0, 4.1]])
scaled_point_y = np.dot(scaling_y, point)

shearing_x = np.array([[1, 7/2], [0, 1]])
sheared_point_x = np.dot(shearing_x, point)

print("Original point: ", point)
print("Reflection through x-axis: ", reflected_point_x)
print("Scaling in x-coordinate by factor 6: ", scaled_point_x)
print("Scaling in y-coordinate by factor 4.1: ", scaled_point_y)
print("Shearing in x-direction by 7/2: ", sheared_point_x)
```

output :

```
Original point:  [-2  4]
Reflection through x-axis:  [-2 -4]
Scaling in x-coordinate by factor 6:  [-12   4]
Scaling in y-coordinate by factor 4.1:  [-2.  16.4]
Shearing in x-direction by 7/2:  [12.  4.]
```

II) Write a python program to find the combined transformation on the line segment between the points A[4,1] & B[-3,0] for the following sequence of transformation

A) Rotation about origin through an angle $\frac{\pi}{4}$

B) Uniform scaling by 7.3 units

C) Scaling in X-coordinate by 3 units

D) Shering in X direction by $\frac{1}{2}$ units

-→

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.transforms import Affine2D
A = np.array([4, 1])
B = np.array([-3, 0])
AB = np.vstack((A, B))

T1 = Affine2D().rotate(np.pi/4) # Rotation about origin through an angle π/4
T2 = Affine2D().scale(7.3, 7.3) # Uniform scaling by 7.3 units
T3 = Affine2D().scale(3, 1) # Scaling in X-coordinate by 3 units
T4 = Affine2D().skew(0.5, 0) # Shering in X direction by 1/2 units

T_combined = T1 + T2 + T3 + T4
AB_transformed = T_combined.transform(AB)
fig, ax = plt.subplots()
ax.plot(AB[:, 0], AB[:, 1], 'b', label='Original')
ax.plot(AB_transformed[:, 0], AB_transformed[:, 1], 'r', label='Transformed')
ax.legend()
ax.set_aspect('equal')
   plt.show()
```