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 6_____Page.no:-_____Class:-_____BCS_____

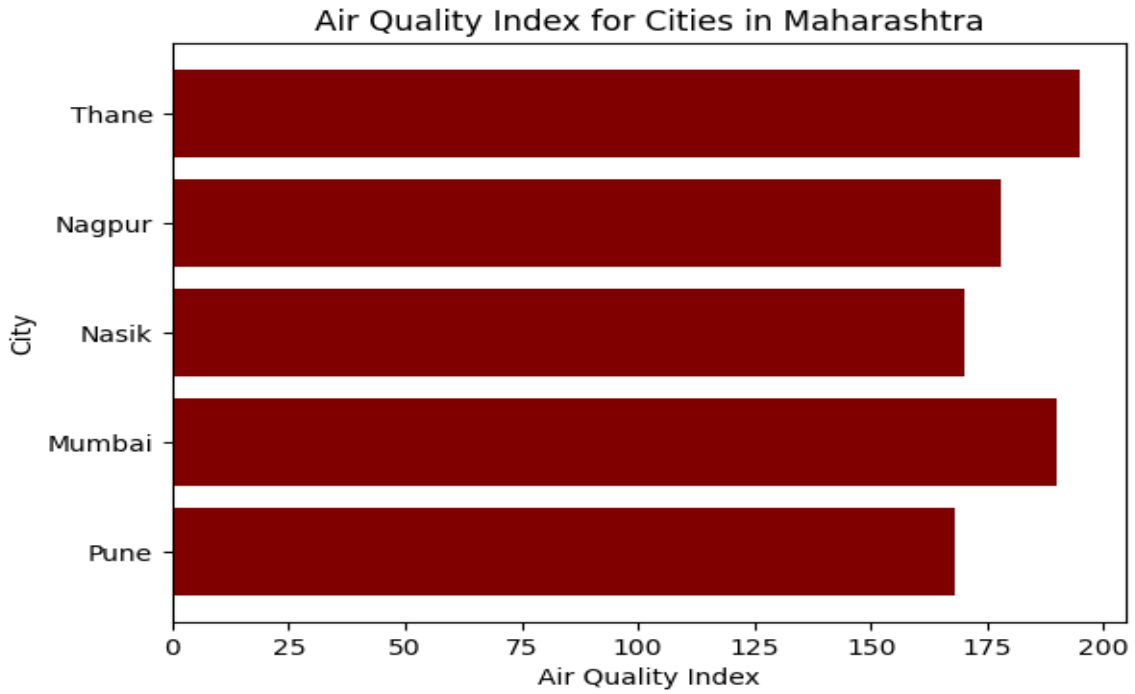
Q1 Attempt any TWO of the following

A) Draw the horizontal bar graph for the following data in Maroon color

City	Pune	Mumbai	Nasik	Nagpur	Thane
Air Quality Idex	168	190	170	178	195

→

```
import matplotlib.pyplot as plt
cities = ['Pune', 'Mumbai', 'Nasik', 'Nagpur', 'Thane']
air_quality = [168, 190, 170, 178, 195]
plt.barh(cities, air_quality, color='maroon')
plt.title('Air Quality Index for Cities in Maharashtra')
plt.xlabel('Air Quality Index')
plt.ylabel('City')
plt.show()
```



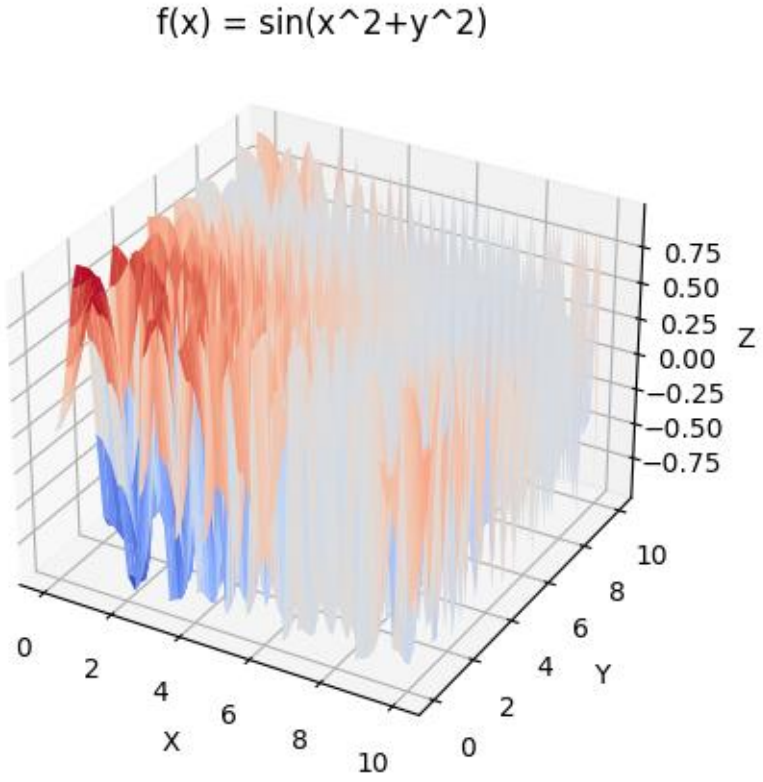
B) Using Python program ,Generate 3D surface plot for the function f(x)=sin($x^{2+}y^2$) in the interval [0,10]

→

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
def f(x, y):
    return np.sin(x**2 + y**2)

x = np.linspace(0, 10, 50)
y = np.linspace(0, 10, 50)
X, Y = np.meshgrid(x, y)
Z = f(X, Y)

fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
ax.plot_surface(X, Y, Z, cmap='coolwarm')
ax.set_title('f(x) = sin(x^2+y^2)')
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Z')
plt.show()
```



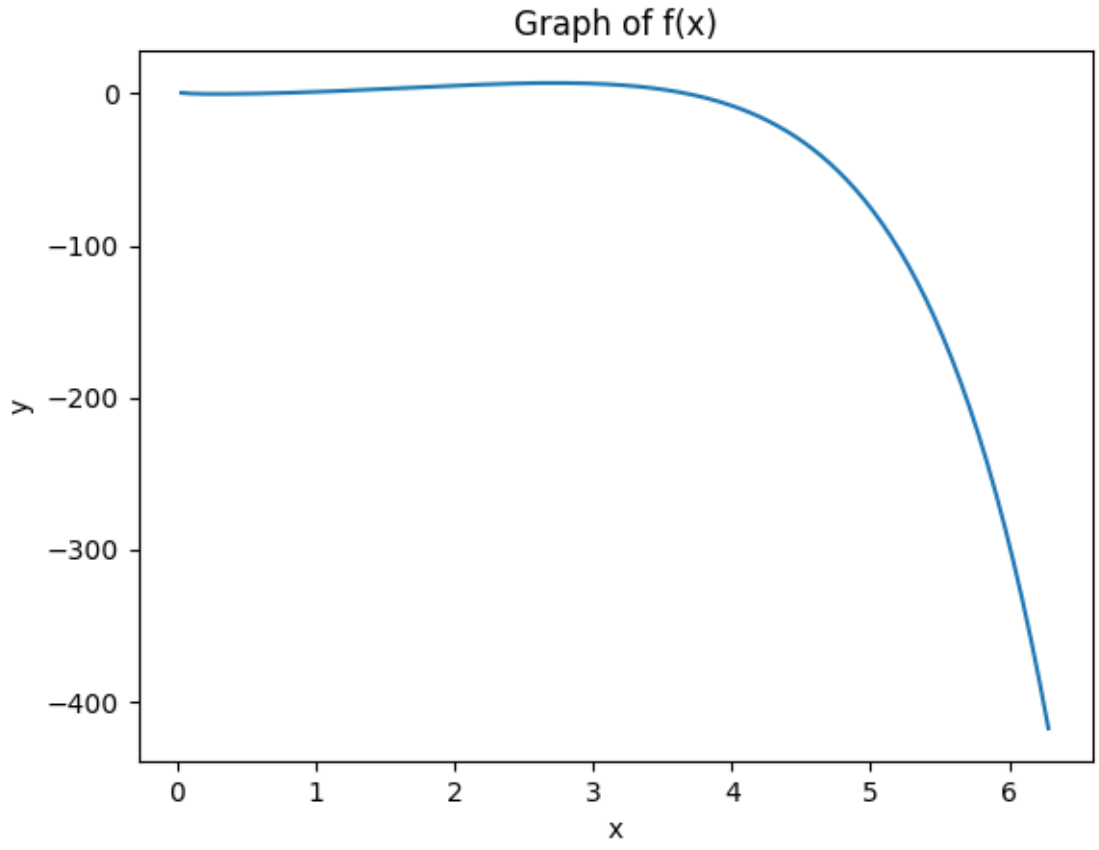
C) Using python ,plot the graph of function $f(x)=\sin(x)-e^{x}+3x^{2}-\log_{10}(X)$ on the interval $[0,2\pi]$

→

```
import numpy as np
import matplotlib.pyplot as plt
def f(x):
    return np.sin(x) - np.exp(x) + 3*x**2 - np.log10(x)

x = np.linspace(0, 2*np.pi, 200)
y = f(x)

plt.plot(x, y)
plt.title('Graph of f(x)')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
```



Q2) Attempt any TWO of the following

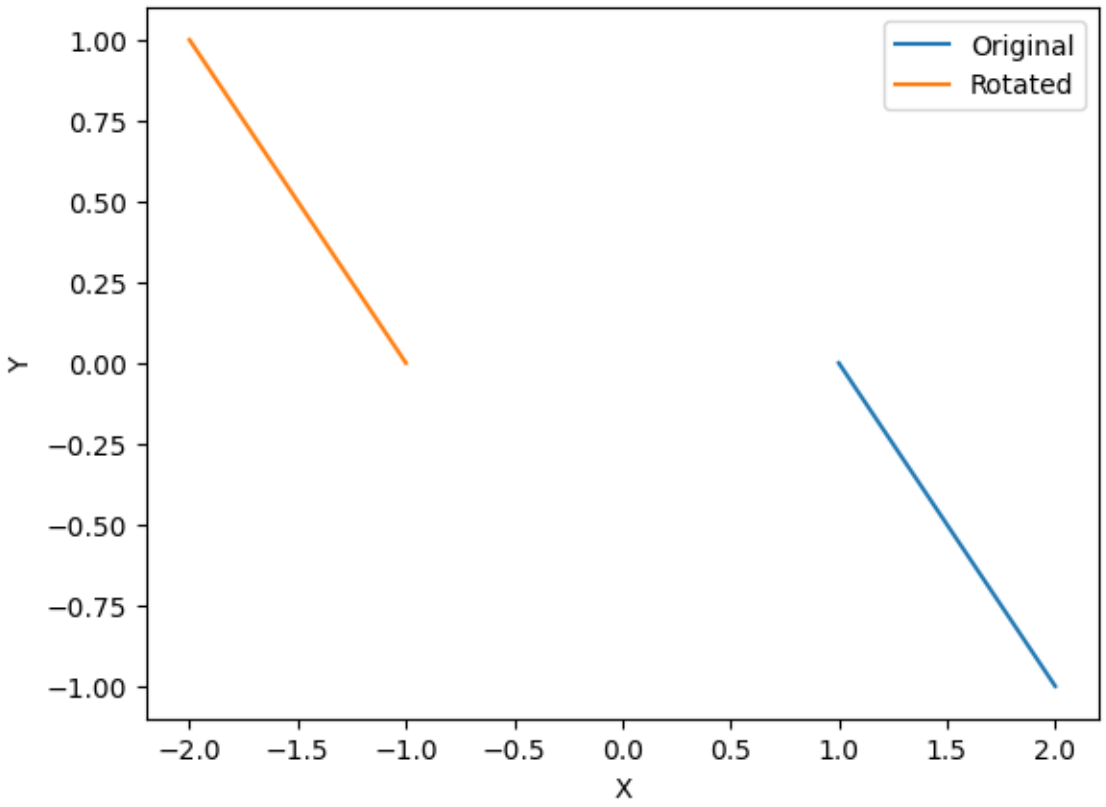
A) Using Python rotate the line segment by 180◦ having end points (1,0) and (2,-1)

-→

```
import numpy as np
import matplotlib.pyplot as plt
x = [1, 2]
y = [0, -1]

theta = np.pi
R = np.array([[np.cos(theta), -np.sin(theta)], [np.sin(theta), np.cos(theta)]])
points = np.vstack((x, y))
rotated_points = R @ points
x_new = rotated_points[0]
y_new = rotated_points[1]

plt.plot(x, y, label='Original')
plt.plot(x_new, y_new, label='Rotated')
plt.xlabel('X')
plt.ylabel('Y')
plt.legend()
plt.show()
```



B) Write a python program , to draw a polygon with vertices (0,0),(2,0),(2,3) and (1,6) and rotate it by using 180◦

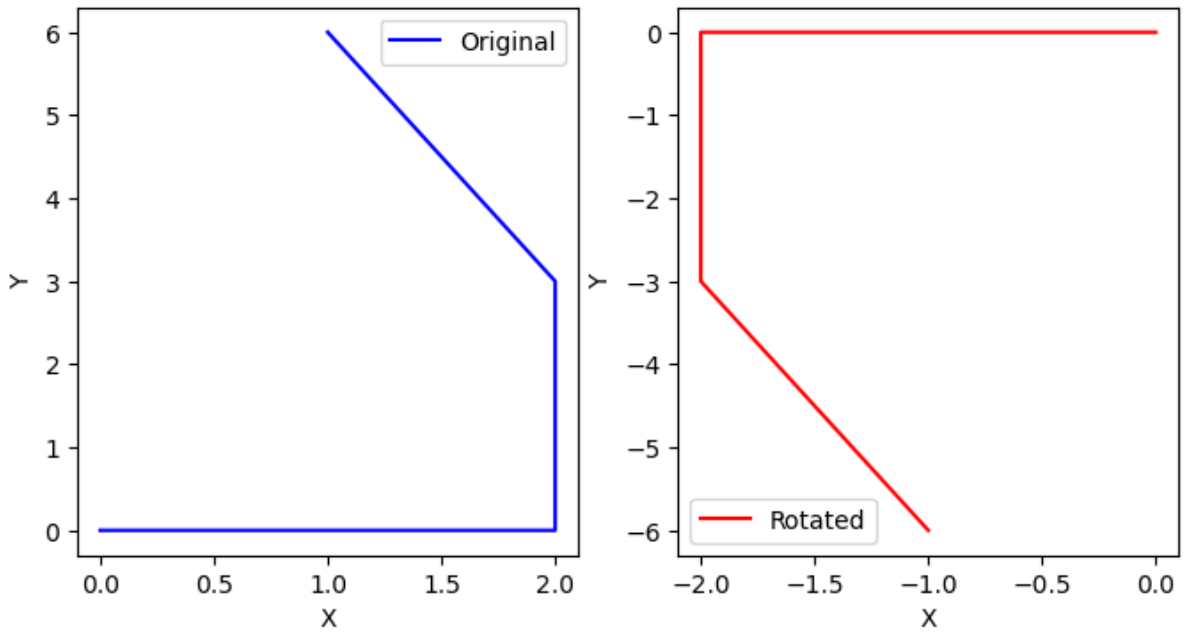
-→

```
  import numpy as np
import matplotlib.pyplot as plt
x = [0, 2, 2, 1]
y = [0, 0, 3, 6]

theta = np.pi
R = np.array([[np.cos(theta), -np.sin(theta)], [np.sin(theta), np.cos(theta)]])
points = np.vstack((x, y))
rotated_points = R @ points
x_new = rotated_points[0]
y_new = rotated_points[1]

fig, axs = plt.subplots(1, 2, figsize=(8, 4))
axs[0].plot(x, y, 'b', label='Original')
axs[1].plot(x_new, y_new, 'r', label='Rotated')

for ax in axs:
    ax.set_xlabel('X')
    ax.set_ylabel('Y')
    ax.legend()
plt.show()
```



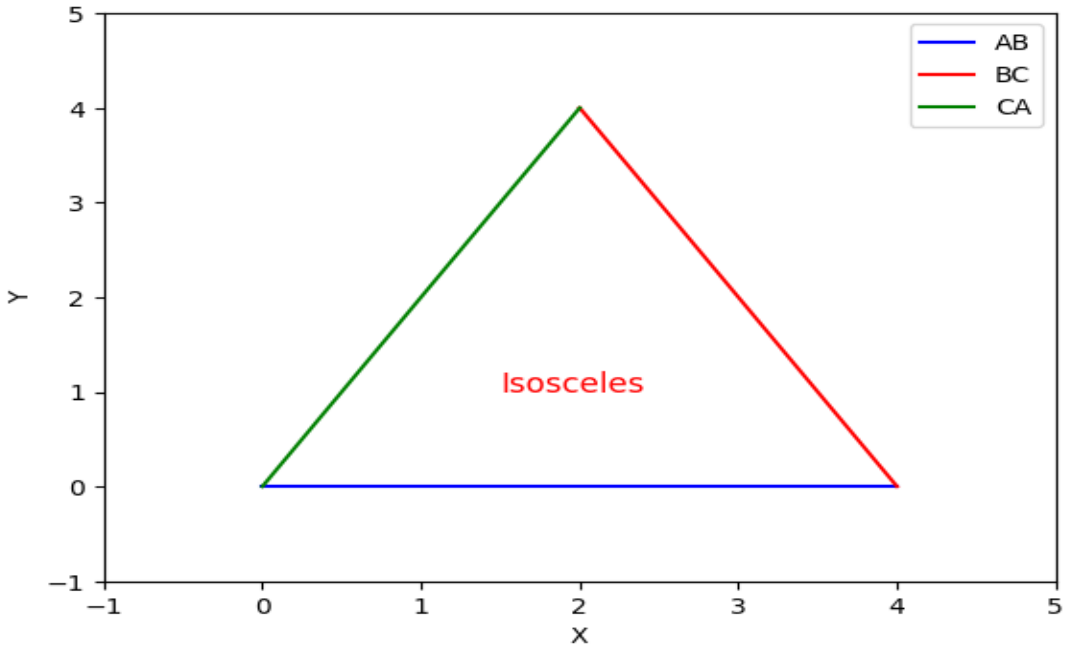
C) Using python program generate tringle with vertices (0,0),(4,0),(2,4) check whether the triangle is isosceles triangle

-→

```
import math
import matplotlib.pyplot as plt
A = (0, 0)
B = (4, 0)
C = (2, 4)
a = math.sqrt((B[0]-C[0])**2 + (B[1]-C[1])**2)
b = math.sqrt((A[0]-C[0])**2 + (A[1]-C[1])**2)
c = math.sqrt((A[0]-B[0])**2 + (A[1]-B[1])**2)

fig, ax = plt.subplots()
ax.plot([A[0], B[0]], [A[1], B[1]], 'b-', label='AB')
ax.plot([B[0], C[0]], [B[1], C[1]], 'r-', label='BC')
ax.plot([C[0], A[0]], [C[1], A[1]], 'g-', label='CA')
if a == b or b == c or a == c:
    ax.text(1.5, 1, 'Isosceles', fontsize=12, color='r')
else:
    ax.text(1.5, 1, 'Non-Isosceles', fontsize=12, color='r')

ax.set_xlim([-1, 5])
ax.set_ylim([-1, 5])
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.legend()
plt.show()
```



Q3) Attempt the following

A) Attempt any one of the following

I) write a python program to solve the following LPP :

Max Z=x+y
Subject to 2x-2y≥1
X+y≥2
X,y≥0

-→

```
from pulp import *
lp_prob = LpProblem("LP problem", LpMaximize)

x = LpVariable('x', lowBound=0, cat='Continuous')
y = LpVariable('y', lowBound=0, cat='Continuous')

lp_prob += x + y
lp_prob += 2*x - 2*y >= 1
lp_prob += x + y >= 2

status = lp_prob.solve()
print("x = ", value(x))
print("y = ", value(y))
print("Optimal objective value = ", value(lp_prob.objective))
```

II) Write a python program to display the following LPP by using Pulp module and simplex method.Find the optimal solution if exist

Min Z=x+y
Subject to x≥6
y≥6
x+y≤11
x,y≥0

-→

```
from pulp import *
lp_prob = LpProblem("LP problem", LpMinimize)
x = LpVariable('x', lowBound=0, cat='Continuous')
y = LpVariable('y', lowBound=0, cat='Continuous')

lp_prob += x + y
lp_prob += x >= 6
lp_prob += y >= 6
lp_prob += x + y <= 11
```

```
status = lp_prob.solve()
print("Status:", LpStatus[status])
if LpStatus[status] == "Optimal":
   print("Optimal solution found:")
   print("x = ", value(x))
   print("y = ", value(y))
   print("Optimal objective value = ", value(lp_prob.objective))
else:
   print("No optimal solution found.")
```

B) Attempt any one of the following

I) Apply Python program in each of the following transformation on the point P[4,-2]

A) Reflection through Y-axis

B) Scaling in X-coordinate By factor 7

C) Shering in Y direction by 3 units

D) Reflection through the line y=-x

-→

```
P = [4, -2]
P_reflected = [-P[0], P[1]]
print("Reflection through Y-axis of P{} is P{}.".format(P, P_reflected))

P = [4, -2]
P_scaled = [7*P[0], P[1]]
print("Scaling in X-coordinate by factor 7 of P{} is P{}.".format(P, P_scaled))

P = [4, -2]
P_sheared = [P[0], P[1] + 3*P[0]]
print("Shearing in Y direction by 3 units of P{} is P{}.".format(P, P_sheared))

P = [4, -2]
P_reflected = [-P[1], -P[0]]
print("Reflection through the line y=-x of P{} is P{}.".format(P, P_reflected))
```

output :

```
Reflection through Y-axis of P[4, -2] is P[-4, -2].
Scaling in X-coordinate by factor 7 of P[4, -2] is P[28, -2].
Shearing in Y direction by 3 units of P[4, -2] is P[4, 10].
Reflection through the line y=-x of P[4, -2] is P[2, -4].
```

II) Find the combined transformation by using Python program for the following sequence of transformation

A) Rotation about origin through an angle 60◦

B) Scaling in X-coordinate by 7 units

C) Uniform scaling by 4 units

D) Reflection through the line y=x

-→

```
import numpy as np
A = np.array([[np.cos(np.radians(60)), -np.sin(np.radians(60)), 0],
              [np.sin(np.radians(60)), np.cos(np.radians(60)), 0],
              [0, 0, 1]])

B = np.array([[7, 0, 0],
              [0, 1, 0],
              [0, 0, 1]])

C = np.array([[4, 0, 0],
              [0, 4, 0],
              [0, 0, 1]])

D = np.array([[0, 1, 0],
              [1, 0, 0],
              [0, 0, 1]])

T = D @ C @ B @ A
p = np.array([[1], [2], [1]])
p_transformed = T @ p
print(p_transformed)
```

output :

```
[[  7.46410162]
 [-34.49742261]
 [  1.        ]]
```