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 11_____Page.no:-_____Class:-_____BCS_____

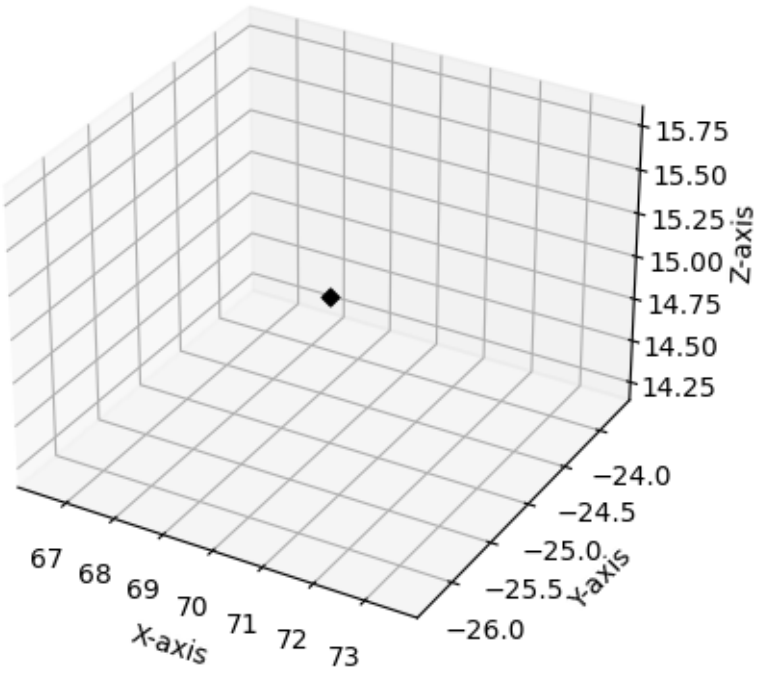
Q1 . Attempt any TWO of the following

A) Write a python program to plot 3D axes with labels as X-axis ,Y-axis and Z-axis and also pack following point with given Coordination in the same graph :(70,-25,15) as a diamond in black color

-→

```
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.set_zlabel('Z-axis')
x, y, z = (70,-25,15)
ax.scatter(x, y, z, marker='D', color='black')
plt.show()
```



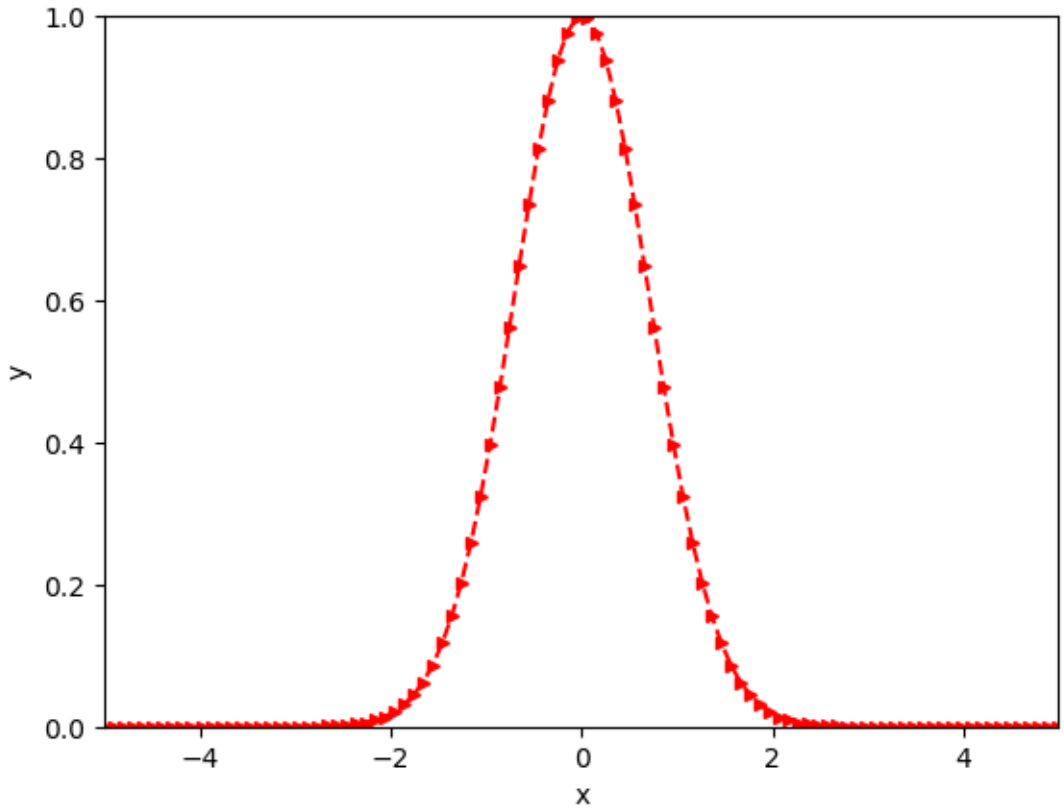
B) Plot the graph of $y=e^{-x^2}$ in [-5,5] with red dashed-points line with Upward Pointing triangle

-→

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

x = np.linspace(-5, 5, 100)
y = f(x)

fig, ax = plt.subplots()
ax.set_xlim([-5, 5])
ax.set_ylim([0, 1])
ax.plot(x, y, 'r-->', markersize=5)
ax.set_xlabel('x')
ax.set_ylabel('y')
plt.show()
```

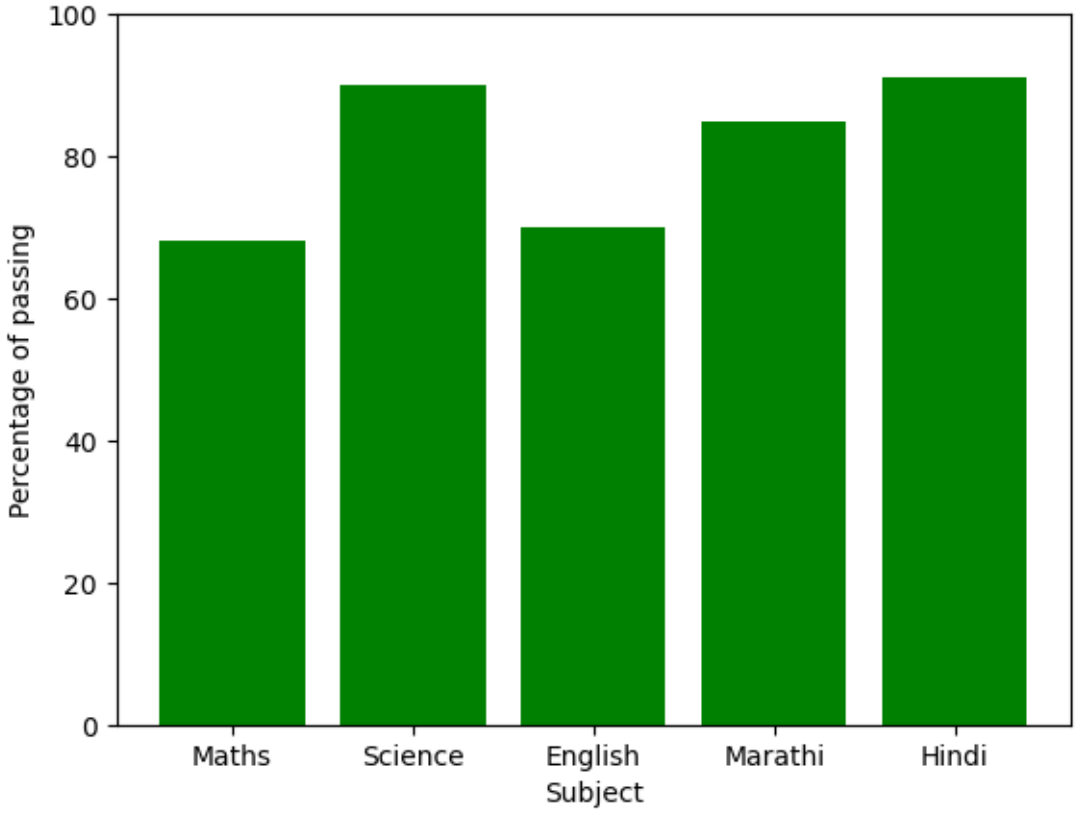


C) Draw a bar graph in GREEN colour to reprenset the data below :

Subject	Maths	Science	English	Marathi	Hindi
Percentage of passing	**68**	**90**	**70**	**85**	**91**

-→

```
import matplotlib.pyplot as plt
subjects = ['Maths', 'Science', 'English', 'Marathi', 'Hindi']
percentages = [68, 90, 70, 85, 91]
fig, ax = plt.subplots()
ax.set_ylim([0, 100])
ax.bar(subjects, percentages, color='green')
ax.set_xlabel('Subject')
ax.set_ylabel('Percentage of passing')
plt.show()
```



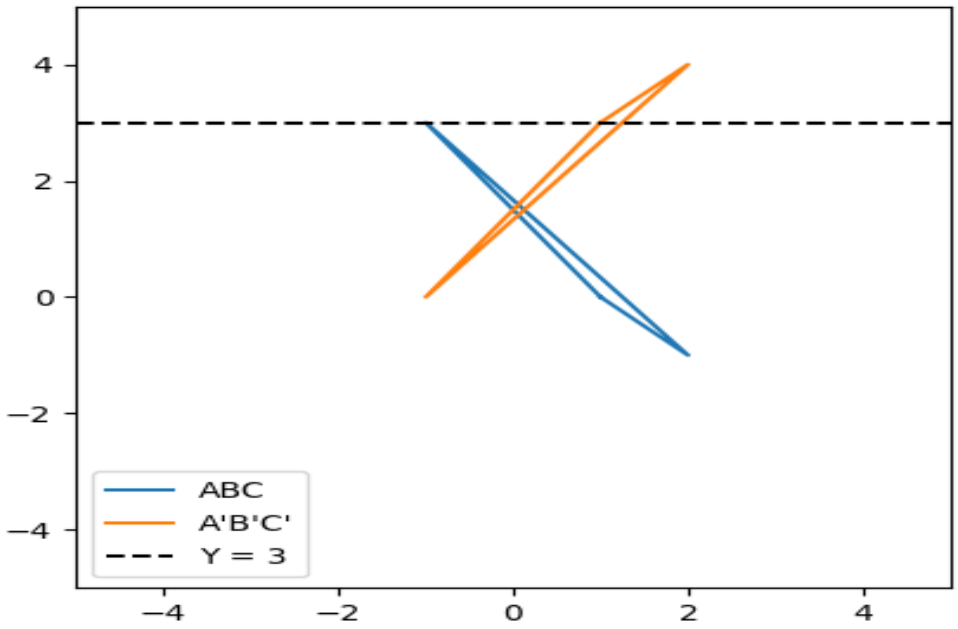
Q2) Attempt any TWO of the following

I) Write a python program to reflect the ΔABC through the line Y=3 where A(1,0),B(2,-1),C(-1,3)

-→

```
import numpy as np
import matplotlib.pyplot as plt
A = np.array([1, 0])
B = np.array([2, -1])
C = np.array([-1, 3])
triangle = np.array([A, B, C, A])
line = np.array([[-5, 3], [5, 3]])
translation_matrix = np.array([[1, 0, 0], [0, 1, -3], [0, 0, 1]])
reflection_matrix = np.array([[1, 0, 0], [0, -1, 0], [0, 0, 1]])
reflected_triangle = np.dot(reflection_matrix, np.concatenate((triangle.T, np.ones((1, 4))), axis=0)).T[:, :2]
inverse_translation_matrix = np.array([[1, 0, 0], [0, 1, 3], [0, 0, 1]])
reflected_triangle = np.dot(inverse_translation_matrix, np.concatenate((reflected_triangle.T, np.ones((1, 4))), axis=0)).T[:, :2]

fig, ax = plt.subplots()
ax.plot(triangle[:, 0], triangle[:, 1], label='ABC')
ax.plot(reflected_triangle[:, 0], reflected_triangle[:, 1], label='A\'B\'C\'')
ax.plot(line[:, 0], line[:, 1], 'k--', label='Y = 3')
ax.set_xlim([-5, 5])
ax.set_ylim([-5, 5])
ax.set_aspect('equal')
ax.legend()
plt.show()
```



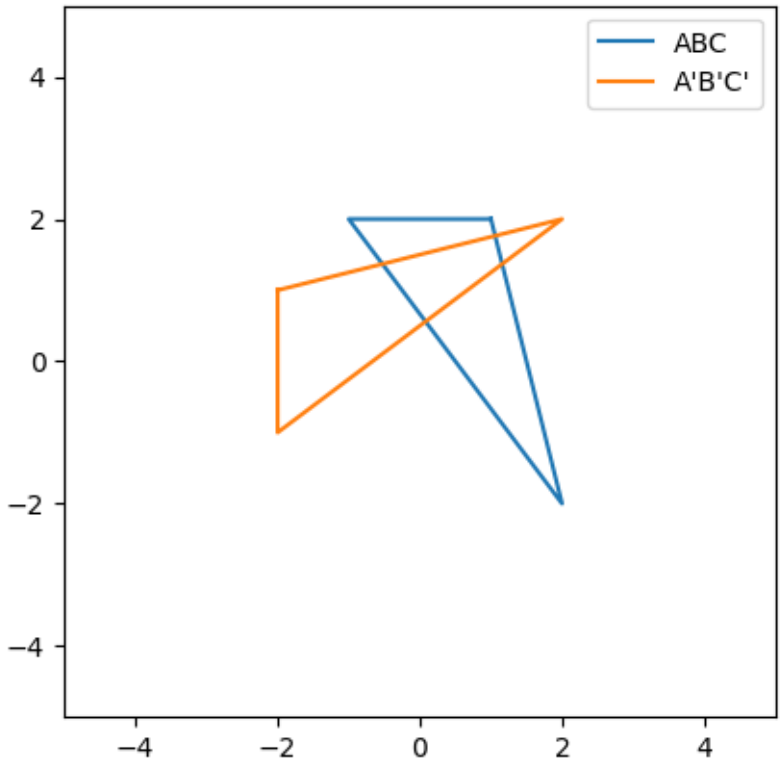
II) Write a python program to rotate the ∆ABC by 90◦ where A(1,2),B(2,-2),C(-1,2)

-→

```
import numpy as np
import matplotlib.pyplot as plt
A = np.array([1, 2])
B = np.array([2, -2])
C = np.array([-1, 2])
triangle = np.array([A, B, C, A])

angle = np.radians(90)
rotation_matrix = np.array([[np.cos(angle), -np.sin(angle)], [np.sin(angle), np.cos(angle)]])
rotated_triangle = np.dot(rotation_matrix, triangle.T).T

fig, ax = plt.subplots()
ax.plot(triangle[:, 0], triangle[:, 1], label='ABC')
ax.plot(rotated_triangle[:, 0], rotated_triangle[:, 1], label='A\'B\'C\'')
ax.set_xlim([-5, 5])
ax.set_ylim([-5, 5])
ax.set_aspect('equal')
ax.legend()
plt.show()
```



III) Write a python program to draw a polygon with 6 sides and radius 1 centered at (1,2) and find its area and perimeter

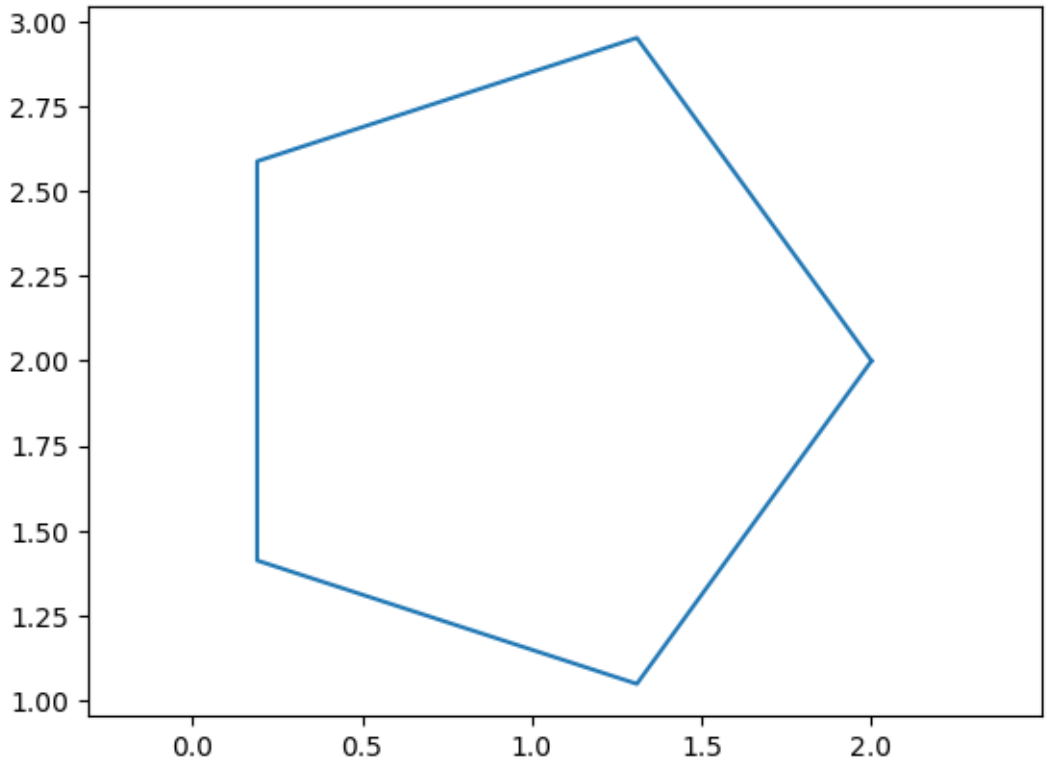
-→

```
import matplotlib.pyplot as plt
import numpy as np
x_center = 1
y_center = 2
n_sides = 6
radius = 1

angles = np.linspace(0, 2 * np.pi, n_sides, endpoint=True)
x = x_center + radius * np.cos(angles)
y = y_center + radius * np.sin(angles)
plt.plot(x, y)
plt.axis('equal')
plt.show()
side_length = 2 * radius * np.sin(np.pi/n_sides)
perimeter = n_sides * side_length
area = (n_sides * radius**2 * np.sin(2 * np.pi / n_sides))/2
print("Perimeter:", perimeter)
print("Area:", area)
```

output :

Perimeter: 5.999999999999999
Area: 2.598076211353316



Q3) Attempt any ONE of the following

I) Attempt any ONE of the following

A) Solve LPP by using Python :

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

-→

```
from scipy.optimize import linprog
c = [1, 1]
A = [[-1, 0], [0, -1], [-1, -1]]
b = [-6, -6, -11]
x_bounds = (0, None)
y_bounds = (0, None)
res = linprog(c, A_ub=A, b_ub=b, bounds=[x_bounds, y_bounds])

print('Optimal value:', res.fun)
print('x:', res.x[0])
print('y:', res.x[1])
```

output :

```
Optimal value: 12.0
x: 6.0
y: 6.0
```

B) Write a python program to solve the following LPP and to find the optimal solution if exists

MAX Z=3x+5y +4z
Subject to 2x+3y≤8
2y+5z≤10
3x+2y+4z≤15
X,y,z≥0

-→

```
import pulp
prob = pulp.LpProblem("LP problem", pulp.LpMaximize)
x = pulp.LpVariable('x', lowBound=0, cat='Continuous')
y = pulp.LpVariable('y', lowBound=0, cat='Continuous')
z = pulp.LpVariable('z', lowBound=0, cat='Continuous')

prob += 3*x + 5*y + 4*z
```

```
prob += 2*x + 3*y <= 8
prob += 2*y + 5*z <= 10
prob += 3*x + 2*y + 4*z <= 15
status = prob.solve()

print("Status:", pulp.LpStatus[status])
print("Optimal Solution:")
print("x =", pulp.value(x))
print("y =", pulp.value(y))
print("z =", pulp.value(z))
print("Optimal Value: Z =", pulp.value(prob.objective))
```

II) Attempt Any one of the following

A) Write a python program to apply the following transformation on the point (-2,4) :
I) Reflection through X-axis
II) Scaling in X-coordinate by factor 6
III) Scaling in x-direction by 4 units
IV) Rotate about origin through an angle 30◦

-→

```
import math
point = [-2, 4]

reflected_point = [point[0], -1*point[1]]
print("I) Reflection through X-axis:", reflected_point)

scaled_point = [6*point[0], point[1]]
print("II) Scaling in X-coordinate by factor 6:", scaled_point)

shifted_point = [point[0] + 4, point[1]]
print("III) Scaling in x-direction by 4 units:", shifted_point)

angle = math.radians(30)
rotated_point = [point[0]*math.cos(angle) - point[1]*math.sin(angle),
                 point[0]*math.sin(angle) + point[1]*math.cos(angle)]
print("IV) Rotate about origin through an angle 30◦:", rotated_point)
```

Output :

I) Reflection through X-axis: [-2, -4]
II) Scaling in X-coordinate by factor 6: [-12, 4]
III) Scaling in x-direction by 4 units: [2, 4]
IV) Rotate about origin through an angle 30◦: [-3.732050807568877, 2.464101615137755]

II)Write a python program to find the combined transformation between the points for the following sequence of transformation

A) Rotation about origin through an angle $\frac{\pi}{2}$

B) uniform scaling by 3.5 units

C) Scaling in X & Y direction by 3 & 5 units respectively

D) shering in X direction by 6 units

-→

```
import math
import numpy as np
point = np.array([2, 5])
theta = math.pi/2
rotation_matrix = np.array([[math.cos(theta), -1*math.sin(theta)],
                            [math.sin(theta), math.cos(theta)]])
point = rotation_matrix.dot(point)
print("After A) Rotation about origin through an angle π/2: ", point)

scaling_factor = 3.5
scaling_matrix = np.array([[scaling_factor, 0],
                           [0, scaling_factor]])
point = scaling_matrix.dot(point)
print("After B) Uniform scaling by 3.5 units: ", point)

scaling_factors = np.array([3, 5])
scaling_matrix = np.diag(scaling_factors)
point = scaling_matrix.dot(point)
print("After C) Scaling in X & Y direction by 3 & 5 units respectively: ", point)

shearing_factor = 6
shearing_matrix = np.array([[1, shearing_factor],
                            [0, 1]])
point = shearing_matrix.dot(point)
print("After D) Shering in X direction by 6 units: ", point)
```

output :

```
After A) Rotation about origin through an angle π/2:  [-5.  2.]
After B) Uniform scaling by 3.5 units:  [-17.5   7. ]
After C) Scaling in X & Y direction by 3 & 5 units respectively:  [-52.5  35. ]
After D) Shering in X direction by 6 units:  [157.5  35. ]
```