PRACTICAL – 5

PROGRAM -1

AIM- WAPP TO PERFORM BOUNDARY FILL ALGORITHM.

CODE

|  |
| --- |
| **import matplotlib.pyplot as plt**  **import numpy as np**  **def boundary\_fill(x, y, fill\_color, boundary\_color, screen):**  **if (x < 0 or x >= len(screen[0]) or y < 0 or y >= len(screen) or**  **screen[y][x] == fill\_color or screen[y][x] == boundary\_color):**  **return**  **screen[y][x] = fill\_color**  **boundary\_fill(x + 1, y, fill\_color, boundary\_color, screen)**  **boundary\_fill(x - 1, y, fill\_color, boundary\_color, screen)**  **boundary\_fill(x, y + 1, fill\_color, boundary\_color, screen)**  **boundary\_fill(x, y - 1, fill\_color, boundary\_color, screen)**  **def create\_plot(screen, colors):**  **height = len(screen)**  **width = len(screen[0])**  **data = np.zeros((height, width, 3), dtype=np.uint8)**  **for y in range(height):**  **for x in range(width):**  **data[y, x] = colors[screen[y][x]]**  **plt.imshow(data)**  **plt.axis('off')**  **plt.show()**  **# Initialize the screen**  **width, height = 10, 10**  **screen = [['blue' for i in range(width)] for j in range(height)]**  **# Define colors**  **colors = {**  **'black': [0, 0, 0],**  **'blue': [0, 0, 255]**  **}**  **# Draw a rectangle**  **for i in range(width):**  **screen[0][i] = 'black'**  **screen[height-1][i] = 'black'**  **for i in range(height):**  **screen[i][0] = 'black'**  **screen[i][width-1] = 'black'**  **# Apply the boundary fill algorithm**  **boundary\_fill(5, 4, 'blue', 'black', screen)**  **# Create and show the plot**  **create\_plot(screen, colors)** |

OUTPUT

|  |
| --- |
|  |