PRACTICAL – 4

PROGRAM -1

AIM- WAPP TO PERFORM FLOOD FILL ALGORITHM.

CODE

|  |
| --- |
| import random  m =[[2,3,3,3,3,3,4,4,4,4,4,4],  [2,3,3,3,3,3,4,4,4,4,4,4],  [3,3,3,1,3,3,4,1,4,4,0,4],  [3,3,1,1,1,3,1,1,1,4,4,4],  [3,1,1,1,1,1,1,1,1,1,4,4],  [3,1,1,0,0,0,0,0,1,1,4,4],  [3,1,1,0,0,0,0,0,1,1,4,4],  [2,1,1,1,1,0,0,0,1,1,2,2],  [2,1,1,1,0,0,0,0,1,2,2,2],  [2,2,1,1,1,0,0,2,2,2,1,0],  [2,2,2,1,1,0,0,2,2,2,1,1],  [2,2,2,2,2,2,2,2,2,0,1,0]  ]  def flood\_recursive(matrix):  width = len(matrix)  height = len(matrix[0])  def ff(x,y,start\_color,color\_to\_update):  #if the square is not the same color as the starting point  if matrix[x][y] != start\_color:  return  #if the square is not the new color  elif matrix[x][y] == color\_to\_update:  return  else:  #update the color of the current square to the replacement color  matrix[x][y] = color\_to\_update  neighbors = [(x-1,y),(x+1,y),(x-1,y-1),(x+1,y+1),(x-1,y+1),(x+1,y-1),(x,y-1),(x,y+1)]  for n in neighbors:  if 0 <= n[0] <= width-1 and 0 <= n[1] <= height-1:  ff(n[0],n[1],start\_color,color\_to\_update)  start\_x = random.randint(0,width-1)  start\_y = random.randint(0,height-1)  start\_color = matrix[start\_x][start\_y]  ff(start\_x,start\_y,start\_color,0)  return matrix  flood\_recursive(m) |

OUTPUT

|  |
| --- |
|  |