Data structures and Algorithms LAB – BSDSF21
(Morning and Afternoon)

Lab 04 – 31-01-2023

1. Implement the following functions recursively and also write their tester main logic.

   void replace(data, height, width, sr, sc, bc, fc)

   Here, the function is similar to bucket filling function of paint software; ***data*** is a 2D array of size ***height*** times ***width*** (have number of rows equal to its height and number of columns equal to its width), ***bc*** is the value at ***sr*** and ***sc*** (the starting point, row and column number within array ***data***) before the replace function called from a main logic function. The function have to recursively replace all connected neighboring elements of array have the same value ***bc*** stored in them with the value ***fc***. The base case may be taken if the neighbors do not contain the same ***bc***, or boundary of the array is crossing.

Following is an illustration of replace(matrix, 8, 9, 3, 5, 2, 11)
Where matrix is declared as **array of 8 rows and 9 columns with data as on below left.**

1	1	1	1	1	**2**	1	1	2
2	2	8	**2**	**2**	**2**	**2**	1	1
2	8	8	**2**	**2**	5	7	8	2
2	8	8	**2**	9	**2**	**2**	8	3
4	4	0	**2**	9	**2**	6	**2**	**2**
0	4	**2**	**2**	9	**2**	**2**	**2**	5
9	4	**2**	**2**	**2**	**2**	**2**	**2**	4
0	4	4	4	4	4	5	4	4
Before replace								

1	1	1	1	1	**11**	1	1	2
2	2	8	**11**	**11**	**11**	**11**	1	1
2	8	8	**11**	**11**	5	7	8	2
2	8	8	**11**	9	**11**	**11**	8	3
4	4	0	**11**	9	**11**	6	**11**	**11**
0	4	**11**	**11**	9	**11**	**11**	**11**	5
9	4	**11**	**11**	**11**	**11**	**11**	**11**	4
0	4	4	4	4	4	5	4	4
After replace								

2. Create a MatrixADT, the abstract data type of Matrix (here is the word Matrix is used in totally different context as in Question 1). Later, implement the following functionalities of the Matrix type:
   - createMatrix(rows, cols) # create empty matrix, you may use __init__ function instead
   - setElement(matrix, rno, cno, val)
   - printMatrix(matrix)
   - sumMatrix(matrix) #return the sum of all values of matrix
   - addMatrices(matrix 1, matrix 2) # return sum of two matrices
   - transpose(matrix) # return the transpose matrix
   - order(matrix) # return the order of matrix
   - isSquare(matrix) # return true if matrix is square matrix
   - isSymmetric(matrix) # return true is matrix is symmetric
   - diagonalSum(matrix) #return the sum of diagonal values

******************* The end ***********