transpose2D

Write a function that takes a square matrix ar, and the array sizes for the rows and columns as parameters, and returns the transpose of the array via call by reference. For example, if the rowSize is 4, colSize is 4, and the array ar is {1,2,3,4, 1,1,2,2, 3,3,4,4, 4,5,6,7}, then the resultant array will be {1,1,3,4, 2,1,3,5, 3,2,4,6, 4,2,4,7}. The function prototype is given below:

```
void transpose2D(int ar[][SIZE], int rowSize, int colSize);
```

A sample program template is given below to test the function:

```
#include <stdio.h>
#define SIZE 10
void transpose2D(int ar[][SIZE], int rowSize, int colSize);
void display(int ar[][SIZE], int rowSize, int colSize);
int main()
{
  int ar[SIZE][SIZE], rowSize, colSize;
  int i,j;

  printf("Enter row size of the 2D array: \n");
  scanf("%d", &rowSize);
  printf("Enter column size of the 2D array: \n");
  scanf("%d", &colSize);
  printf("Enter the matrix (%dx%d): \n", rowSize, colSize);
  for (i=0; i<rowSize; i++)
    for (j=0; j<colSize; j++)
      scanf("%d", &ar[i][j]);
  printf("transpose2D(): \n");
  transpose2D(ar, rowSize, colSize);
  display(ar, rowSize, colSize);
  return 0;
}
void display(int ar[][SIZE], int rowSize, int colSize)
{
  int l,m;
  for (l = 0; l < rowSize; l++) {
    for (m = 0; m < colSize; m++)
      printf("%d ", ar[l][m]);
    printf("\n");
  }
}
void transpose2D(int ar[][SIZE], int rowSize, int colSize)
{
     /* Write your program code here */
}
```

Some sample input and output sessions are given below:

(1) Test Case 1:
Enter row size of the 2D array:

4
Enter column size of the 2D array:
4
Enter the matrix (4x4):
1 2 3 4
1 1 2 2
3 3 4 4
4 5 6 7
transpose2D():
1 1 3 4
2 1 3 5
3 2 4 6
4 2 4 7

(2) Test Case 2:
Enter row size of the 2D array:
3
Enter column size of the 2D array:
3
Enter the matrix (3x3):
1 2 3
3 4 5
5 6 7
transpose2D():
1 3 5
2 4 6
3 5 7

(3) Test Case 3:
Enter row size of the 2D array:
5
Enter column size of the 2D array:
5
Enter the matrix (4x4):
1 2 3 4 5
1 1 2 2 5
3 3 4 4 5
4 5 6 7 5
1 1 2 2 5
transpose2D():
1 1 3 4 1
2 1 3 5 1
3 2 4 6 2
4 2 4 7 2
5 5 5 5 5