reduceMatrix2D

A square matrix (2-dimensional array of equal dimensions) can be reduced to upper-triangular form by setting each diagonal element to the sum of the original elements in that column and setting to 0s all the elements below the diagonal. For example, the 4-by-4 matrix:

$$\begin{matrix} 4 & 3 & 8 & 6 \\ 9 & 0 & 6 & 5 \\ 5 & 1 & 2 & 4 \\ 9 & 8 & 3 & 7 \end{matrix}$$

would be reduced to

$$\begin{matrix} 27 & 3 & 8 & 6 \\ 0 & 9 & 6 & 5 \\ 0 & 0 & 5 & 4 \\ 0 & 0 & 0 & 7 \end{matrix}$$

Write a function reduceMatrix2D() to reduce a matrix with dimensions of *rowSize* and *colSize*. The prototype of the function is:

```
void reduceMatrix2D(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 reduceMatrix2D(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]);
  reduceMatrix2D(ar, rowSize, colSize);
  printf("reduceMatrix2D(): \n");
  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 reduceMatrix2D(int ar[][SIZE], int rowSize, int colSize)
{
  /* Write your 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
5 6 7 8
9 10 11 12
13 14 15 16
reduceMatrix2D():
28 2 3 4
0 30 7 8
0 0 26 12
0 0 0 16
```

(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 0 0
2 2 0
3 3 3
reduceMatrix2D():
6 0 0
0 5 0
0 0 3
```

(3) Test Case 3:

```
Enter row size of the 2D array:
4
Enter column size of the 2D array:
4
Enter the matrix (4x4):
1 2 3 4
7 8 9 10
5 6 7 8
11 12 13 14
```

```
reduceMatrix2D():
24 2 3 4
0 26 9 10
0 0 20 8
0 0 0 14
```

(4) Test Case 4:

```
Enter row size of the 2D array:
4
Enter column size of the 2D array:
4
Enter the matrix (4x4):
-5 -6 -7 -8
3 4 5 6
-1 -2 -3 -4
6 7 8 9
reduceMatrix2D():
3 -6 -7 -8
0 9 5 6
0 0 5 -4
0 0 0 9
```