7. Jeremy

Program Name: Jeremy.java **Input File: jeremy.dat**

Jeremy knows that bitmap images can be represented by a matrix of integers, with each integer in the matrix representing the color of a "pixel". Various editing operations can be performed on a bitmap, but one of the most common ones is the flood fill. He knows that a flood fill is a change process that starts at a single pixel, changing every adjacent pixel that is the same color as the starting pixel *(not including adjacent cells in diagonal directions)*, to another color. The process continues for all the pixels that were changed, until an entire block of color in the picture has been changed. Jeremy needs your help in creating a program that, given the length and width of a bitmap, a matrix of integers from 0-9 that represents the bitmap, the starting pixel, and a "color" from 0-9 to change to, will perform a flood fill operation on that bitmap.

For example, in this 4 x 7 matrix, the color at position (2,2) is a 4, as shown in bold. If this location was changed to the color 6, and then a flood fill is performed for all neighboring values of 4, resulting in the second matrix. The remaining 4 in the top right corner stays unchanged since it is not reachable.

```
0 3 4 4 2 9 4
4 2 4 3 2 1 8
4 4 4 4 3 5 6
2 0 4 4 4 4 5

0 3 6 6 2 9 4
6 2 6 3 2 1 8
6 6 6 6 3 5 6
2 0 6 6 6 6 5
```

Input: An initial integer N, representing N data sets to follow. Each data set consists of two integers R and C, indicating an R x C matrix of integers, which follows on the next R rows. The matrix consists of single integers in the range 0-9, with single space separation. Following the matrix are two integers A and B representing the target location in the matrix, followed by an integer D as the flood fill color.

Output: The resulting matrix after the flood fill operation, which changes all instances of the color integer at location (A,B) to the color integer D, as described above and shown in the examples below. Print a final "**=====**" line below each complete output.

Sample input:

```
2
4 7
0 3 4 4 2 9 4
4 2 4 3 2 1 8
4 4 4 4 3 5 6
2 0 4 4 4 4 5
2 2
6
5 8
0 0 0 0 0 0 1 1
0 0 1 1 1 1 2 2
0 1 1 2 2 2 2 3
0 1 2 3 2 3 4 5
1 2 3 4 2 6 2 8
1 7
9
```

Sample output:

```
0 3 6 6 2 9 4
6 2 6 3 2 1 8
6 6 6 6 3 5 6
2 0 6 6 6 6 5
=====
0 0 0 0 0 0 1 1
0 0 1 1 1 1 9 9
0 1 1 9 9 9 9 3
0 1 2 3 9 3 4 5
1 2 3 4 9 6 2 8
=====
```