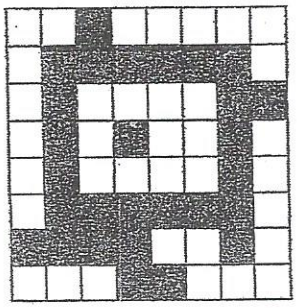
Example

On the right is an image represented as a square grid of black and white cells. Two cells in an image are part of the same "blob" if each is black and there is a sequence of moves from one cell to the other, where each move is either horizontal or vertical to an adjacent black cell. For example, the diagram represents an image that contains two blobs, one of them consisting of a single cell.



Assuming the following `Image` class declaration, you are to write the body of the `eraseBlob` method, using a recursive algorithm.

```
public class Image
{
    private final int BLACK = 1;
    private final int WHITE = 0;
    private int[][] image;    //square grid
    private int size;         //number of rows and columns

    public Image()   //constructor
    { /* implementation not shown */ }

    public void display()   //displays Image
    { /* implementation not shown */ }

    /* Precondition:  Image is defined with either BLACK or WHITE
                      cells.
     * Postcondition: If 0 <= row < size, 0 <= col < size,
     *                and image[row][col] is BLACK, set all cells
     *                in the same blob to WHITE. Otherwise image
     *                is unchanged. */
    public void eraseBlob(int row, int col)
    /* your code goes here */
}
```

Solution:

```
public void eraseBlob(int row, int col)
{
    if (row >= 0 && row < size && col >= 0 && col < size)
        if (image[row][col] == BLACK)
        {
            image[row][col] = WHITE;
            eraseBlob(row - 1, col);
            eraseBlob(row + 1, col);
            eraseBlob(row, col - 1);
            eraseBlob(row, col + 1);
        }
}
```

NOTE

1. The ordering of the four recursive calls is irrelevant.

2. The test

```
if (image[row][col] == BLACK)
```

can be included as the last piece of the test in the first line:

```
if (row >= 0 && ...
```

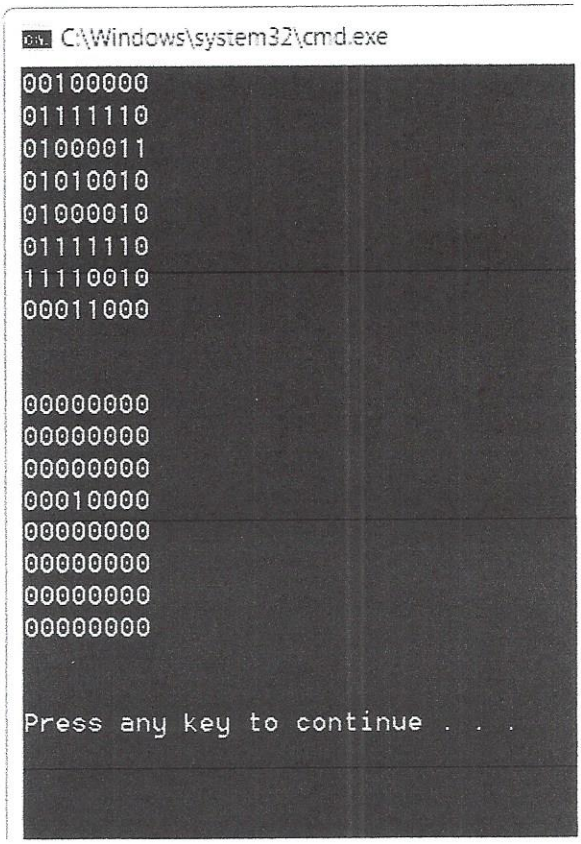
If `row` or `col` is out of range, the test will short-circuit, avoiding the dreaded `ArrayIndexOutOfBoundsException`.

3. If you put the statement

```
image[row][col] = WHITE;
```

after the four recursive calls, you get infinite recursion if your blob has more than one cell. This is because, when you visit an adjacent cell, one of its recursive calls visits the original cell. If this cell is still `BLACK`, yet more recursive calls are generated, *ad infinitum*.

A final thought: Recursive algorithms can be tricky. Try to state the solution recursively *in words* before you launch into code. Oh, and don't forget the base case!


C:\Windows\system32\cmd.exe
00100000
01111110
01000011
01010010
01000010
01111110
11110010
00011000
00000000
00000000
00000000
00010000
00000000
00000000
00000000
00000000
Press any key to continue . . .