```
def ExploreAndLabelColony(grid, x, y, label):

    grid[x][y] = label                      //REPLACE CURRENT INDEX WITH A LABEL
    colonySize = colonySize + 1             //INCREASE COLONY SIZE BY 1

//CHECK TOP LEFT CORNER
if ((x NOT 0 AND y NOT 0) AND (grid[x - 1][y - 1] == '1'))
        ExploreAndLabelColony(grid, x - 1, y - 1, label)

//CHECK TOP
if ((x NOT 0 AND grid[x - 1][y] == '1'))
        ExploreAndLabelColony(grid, x - 1, y, label)

//CHECK TOP RIGHT CORNER
if ((x NOT 0 AND y NOT (grid[x].length - 1)) AND (grid[x - 1][y + 1] == '1'))
        ExploreAndLabelColony(grid, x - 1, y + 1, label)

// RIGHT
if ((y NOT (grid[x].length - 1)) AND (grid[x][y + 1] == '1'))
        ExploreAndLabelColony(grid, x, y + 1, label)

// BOTTOM RIGHT CORNER
if ((x NOT (grid.length - 1) AND y NOT (grid[x].length - 1) ) AND (grid[x + 1][y + 1] == '1'))
        ExploreAndLabelColony(grid, x + 1, y + 1, label)

// BOTTOM
if ((x NOT (grid.length - 1)) AND (grid[x + 1][y] == '1'))
        ExploreAndLabelColony(grid, x + 1, y, label)

// BOTTOM LEFT CORNER
if ((y NOT 0 AND x NOT (grid.length - 1)) AND (grid[x + 1][y - 1] == '1'))
        ExploreAndLabelColony(grid, x + 1, y - 1, label)

// LEFT
if ((y NOT 0) AND (grid[x][y - 1] == '1'))
        ExploreAndLabelColony(grid, x, y - 1, label)

//RETURN THE SIZE OF THE CURRENT COLONY
return(colonySize)
```

Parameter	Type	Explanation
grid	2D Array of 1’s and 0’s	Grid containing all 1’s and 0’s
x	Integer	row number in the 2D Array
y	Integer	Column number in the 2D Array
Label	Character	The label to convert 1’s to

Output	Type	Explanation
colonySize	Integer	The number of 1’s changed to labels

The worst case scenario is when we have a m*n array filled with 1’s (ie: a single colony). This situation will result in the most number of recursive calls being made and so the recursive call stack will fill up the most in this case.

For each 1 that is found in the grid, a recursive call is made.
If for example we have the following grid:

1	1	1
1	1	1
1	1	1

and the following function call was made ExploreAndLabelColony(grid, 1, 1, ‘A’) then our recursive call stack would be as follows:

Left 1
Bottom Left 1
Bottom 1
Bottom Right 1
Right 1
Top Right 1
Top 1
Top Left 1
Initial Call

ExploreAndLabelColony(grid, 1, 0, ‘A’)
ExploreAndLabelColony(grid, 2, 0, ‘A’)
ExploreAndLabelColony(grid, 2, 1, ‘A’)
ExploreAndLabelColony(grid, 2, 2, ‘A’)
ExploreAndLabelColony(grid, 1, 2, ‘A’)
ExploreAndLabelColony(grid, 0, 2, ‘A’)
ExploreAndLabelColony(grid, 0, 1, ‘A’)
ExploreAndLabelColony(grid, 0, 0, ‘A’)
ExploreAndLabelColony(grid, 1, 1, ‘A’)

We see that the maximum size of the recursive call stack is the number of 1’s present in the grid. Therefore, in the worst case we have m*n number of 1’s in the grid. There is also a constant amount of work done during each function call, say c.

We can model the time function as follows:
(rows: m, columns: n, constant work done: c)

$$(m * n) + c$$

Therefore the time complexity of the recursive method is:

$$\mathbf{O(m * n)}$$

Similarly the space complexity of the recursive method is:

$$\mathbf{O(m * n)}$$

Since the maximum size of the recursive call stack is the number of 1's – which in the worst case is m*n

The type of recursion used for this method is **Tree Recursion**.
This is because there are multiple recursive calls being made in a single invocation of the recursive method.

A tail recursive solution is also possible.
(See ColonyExplorerRecursiveTail.java for more)
(Pseudocode shown on next page)

Recursive Solution – Programming Section

Zafir Khalid - 40152164
Marwa Khalid - 40155098

```
def ExploreAndLabelColony(grid, x, y, label, index):

        if(grid[x][y] == '1'):
                grid[x][y] = label
                INCREMENT colonySize

        INITIALIZE ArrayList TO STORE SURRONDING COORDINATES

        while(index < (rows*cols)):

        if (x NOT 0 AND y NOT 0 AND grid[x - 1][y - 1] == '1'):
                surrondings.add(x-1)
                surrondings.add(y-1)

        if (x NOT 0 AND grid[x - 1][y] == '1'):
                surrondings.add(x-1)
                surrondings.add(y)

        if (x NOT 0 AND y NOT (grid[x].length - 1) AND grid[x - 1][y + 1] == '1'):
                surrondings.add(x-1)
                surrondings.add(y+1)

        if (y NOT (grid[x].length - 1) AND grid[x][y + 1] == '1'):
                surrondings.add(x)
                surrondings.add(y+1)

        if (x NOT (grid.length - 1) AND y NOT (grid[x].length - 1) AND grid[x + 1][y + 1] == '1'):
                surrondings.add(x+1)
                surrondings.add(y+1)

        if (x NOT (grid.length - 1) AND grid[x + 1][y] == '1'):
                surrondings.add(x+1)
                surrondings.add(y)

        if (y NOT 0 AND x NOT (grid.length - 1) AND grid[x + 1][y - 1] == '1'):
                surrondings.add(x+1)
                surrondings.add(y-1)

        if (y NOT 0 AND grid[x][y - 1] == '1'):
                surrondings.add(x)
                surrondings.add(y-1)

        index = index + 1

        FOR k : 0 to surrondings.size()/2
                newX = surrondings.get(k)
                newY = surrondings.get(k+1)
                ExploreAndLabelColonyTail(newX, newY, grid, label, 0)
```