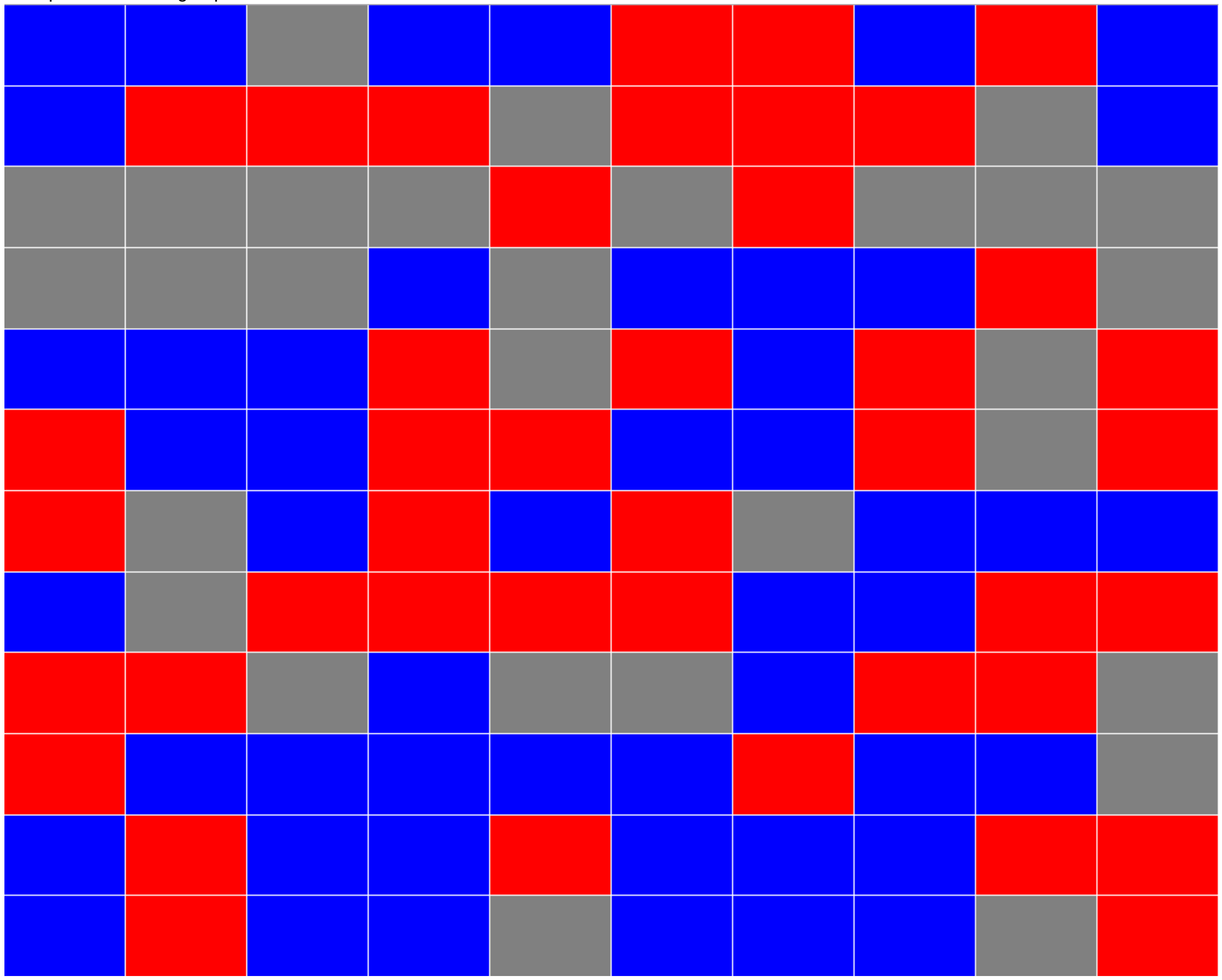
Continuous Color Grid Count Challenge

In this test, you need to write a program to analyse a given color grid, and find out the largest continuous color block in the given grid. Here is an example of the color grid problem.



As shown in the picture above, a color grid is a two-dimensional board formed by color nodes. Here are ***some Facts*** about the color grid:

- Each color grid consists of **Rows** and **Columns**, where the rows and columns can be uneven
- Each **Node** in the color grid has its own color (e.g. Blue, Red, Green, etc)
- Each node can have a number of **Neighbour/Adjacent Nodes**. Depend on its location/coordinate, a node can have up to 4 neighbour nodes, or 3 neighour nodes if it's on the side, or two neighbour nodes if it's in the corner.
- The color grid can have **N** number of colors, and **X** number of rows, as well as **Y** number of columns. **N, X, Y** are unknown in the problem, and is not given at the beginning of the test

The challenge: since each color node can be adjacent to a number of other color nodes, you need to find out the largest connecting block of nodes with the same color. (In the above image, by looking at the image we know that the largest block of joining nodes is at the bottom part of the image with blue color)

Solution:

- You can choose the programming language of your choice
- You can use any types of data model to store the grid information
- You can use any algorithm in your program to conduct the search
- The solution will be evaluated based on its accuracy and speed

Tips:

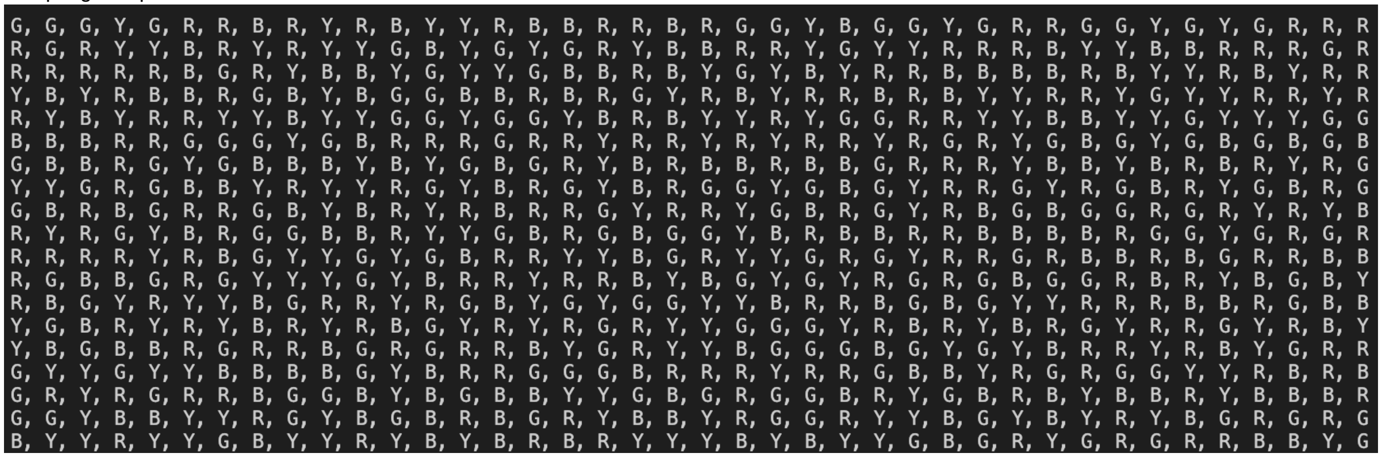
- Start with designing the right data model to store the color grid and search result
- Try avoid using recursion for large color grid, however, recursion can be used in some cases with careful design

Sample Code:

The following sample code generates a random color grid, given its col and row size.

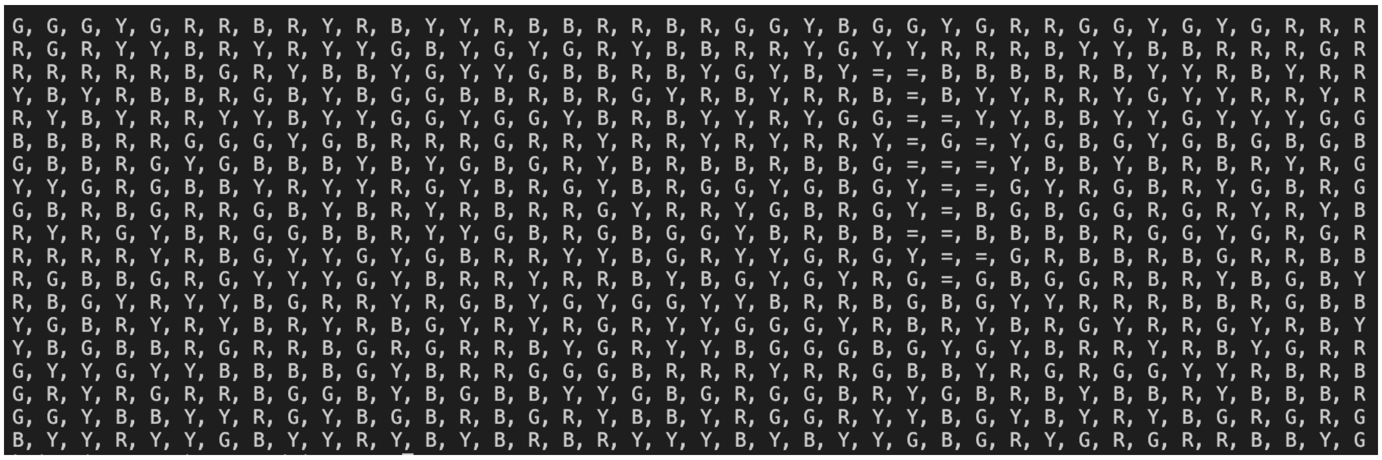
```
function createGame(col, row) {
    return Array(col*row).fill()
        .map(
            (
            item,
            index,
            ) => ({
            color: random_color, //Random color code
            x: index % col,
            y: Math.floor(index / col),
            })
        )
}

function printGame(game) {
    var line = ""
    game.forEach(
        n => {
            if(n.x == 0) {
                console.log(line)
                line = ""
            }
            line += n.color + (n.x % COL_NUM == (COL_NUM-1) ? " " : ",
")
        }
    )
}
```

Sample game printed on console



```
G, G, G, Y, G, R, R, B, R, Y, R, B, Y, Y, R, B, B, R, R, B, R, G, G, Y, B, G, G, Y, G, R, R, G, G, Y, G, Y, G, R, R, R
R, G, R, Y, Y, B, R, Y, R, Y, Y, G, B, Y, G, Y, G, R, Y, B, B, R, R, Y, G, Y, Y, R, R, R, B, Y, Y, B, B, R, R, R, G, R
R, R, R, R, R, B, G, R, Y, B, B, Y, G, Y, Y, G, B, B, R, B, Y, G, Y, B, Y, R, R, B, B, B, B, R, B, Y, Y, R, B, Y, R, R
Y, B, Y, R, B, B, R, G, B, Y, B, G, G, B, B, R, B, R, G, Y, R, B, Y, R, R, B, R, B, Y, Y, R, R, Y, G, Y, Y, R, R, Y, R
R, Y, B, Y, R, R, Y, Y, B, Y, Y, G, G, Y, G, G, Y, B, R, B, Y, Y, R, Y, G, G, R, R, Y, Y, B, B, Y, Y, G, Y, Y, Y, G, G
B, B, B, R, R, G, G, G, Y, G, B, R, R, R, G, R, R, Y, R, R, Y, R, Y, R, R, Y, R, G, R, Y, G, B, G, Y, G, B, G, B, G, B
G, B, B, R, G, Y, G, B, B, B, Y, B, Y, G, B, G, R, Y, B, R, B, B, R, B, B, G, R, R, R, Y, B, B, Y, B, R, B, R, Y, R, G
Y, Y, G, R, G, B, B, Y, R, Y, Y, R, G, Y, B, R, G, Y, B, R, G, G, Y, G, B, G, Y, R, R, G, Y, R, G, B, R, Y, G, B, R, G
G, B, R, B, G, R, R, G, B, Y, B, R, Y, R, B, R, R, G, Y, R, R, Y, G, B, R, G, Y, R, B, G, B, G, G, R, G, R, Y, R, Y, B
R, Y, R, G, Y, B, R, G, G, B, B, R, Y, Y, G, B, R, G, B, G, G, Y, B, R, B, B, R, R, B, B, B, B, R, G, G, Y, G, R, G, R
R, R, R, R, Y, R, B, G, Y, Y, G, Y, G, B, R, R, Y, Y, B, G, R, Y, Y, G, R, G, Y, R, R, G, R, B, B, R, B, G, R, R, B, B
R, G, B, B, G, R, G, Y, Y, Y, G, Y, B, R, R, Y, R, R, B, Y, B, G, Y, G, Y, R, G, R, G, B, G, G, R, B, R, Y, B, G, B, Y
R, B, G, Y, R, Y, Y, B, G, R, R, Y, R, G, B, Y, G, Y, G, G, Y, Y, B, R, R, B, G, B, G, Y, Y, R, R, R, B, B, R, G, B, B
Y, G, B, R, Y, R, Y, B, R, Y, R, B, G, Y, R, Y, R, G, R, Y, Y, G, G, G, Y, R, B, R, Y, B, R, G, Y, R, R, G, Y, R, B, Y
Y, B, G, B, B, R, G, R, R, B, G, R, G, R, R, B, Y, G, R, Y, Y, B, G, G, G, B, G, Y, G, Y, B, R, R, Y, R, B, Y, G, R, R
G, Y, Y, G, Y, Y, B, B, B, B, G, Y, B, R, R, G, G, G, B, R, R, R, Y, R, R, G, B, B, Y, R, G, R, G, G, Y, Y, R, B, R, B
G, R, Y, R, G, R, R, B, G, G, B, Y, B, G, B, B, Y, Y, G, B, G, R, G, G, B, R, Y, G, B, R, B, Y, B, B, R, Y, B, B, B, R
G, G, Y, B, B, Y, Y, R, G, Y, B, G, B, R, B, G, R, Y, B, B, Y, R, G, G, R, Y, Y, B, G, Y, B, Y, R, Y, B, G, R, G, R, G
B, Y, Y, R, Y, Y, G, B, Y, Y, R, Y, B, Y, B, R, B, R, Y, Y, Y, B, Y, B, Y, Y, G, B, G, R, Y, G, R, G, R, R, B, B, Y, G
```

Sample game with largest continous block (LCB) marked by “=“ symbol



```
G, G, G, Y, G, R, R, B, R, Y, R, B, Y, Y, R, B, B, R, R, B, R, G, G, Y, B, G, G, Y, G, R, R, G, G, Y, G, Y, G, R, R, R
R, G, R, Y, Y, B, R, Y, R, Y, Y, G, B, Y, G, Y, G, R, Y, B, B, R, R, Y, G, Y, Y, R, R, R, B, Y, Y, B, B, R, R, R, G, R
R, R, R, R, R, B, G, R, Y, B, B, Y, G, Y, Y, G, B, B, R, B, Y, G, Y, B, Y, =, =, B, B, B, B, R, B, Y, Y, R, B, Y, R, R
Y, B, Y, R, B, B, R, G, B, Y, B, G, G, B, B, R, B, R, G, Y, R, B, Y, R, R, B, =, B, Y, Y, R, R, Y, G, Y, Y, R, R, Y, R
R, Y, B, Y, R, R, Y, Y, B, Y, Y, G, G, Y, G, G, Y, B, R, B, Y, Y, R, Y, G, G, =, =, Y, Y, B, B, Y, Y, G, Y, Y, Y, G, G
B, B, B, R, R, G, G, G, Y, G, B, R, R, R, G, R, R, Y, R, R, Y, R, Y, R, R, Y, =, G, =, Y, G, B, G, Y, G, B, G, B, G, B
G, B, B, R, G, Y, G, B, B, B, Y, B, Y, G, B, G, R, Y, B, R, B, B, R, B, B, G, =, =, =, Y, B, B, Y, B, R, B, R, Y, R, G
Y, Y, G, R, G, B, B, Y, R, Y, Y, R, G, Y, B, R, G, Y, B, R, G, G, Y, G, B, G, Y, =, =, G, Y, R, G, B, R, Y, G, B, R, G
G, B, R, B, G, R, R, G, B, Y, B, R, Y, R, B, R, R, G, Y, R, R, Y, G, B, R, G, Y, =, B, G, B, G, G, R, G, R, Y, R, Y, B
R, Y, R, G, Y, B, R, G, G, B, B, R, Y, Y, G, B, R, G, B, G, G, Y, B, R, B, B, =, =, B, B, B, B, R, G, G, Y, G, R, G, R
R, R, R, R, Y, R, B, G, Y, Y, G, Y, G, B, R, R, Y, Y, B, G, R, Y, Y, G, R, G, Y, =, =, G, R, B, B, R, B, G, R, R, B, B
R, G, B, B, G, R, G, Y, Y, Y, G, Y, B, R, R, Y, R, R, B, Y, B, G, Y, G, Y, R, G, =, G, B, G, G, R, B, R, Y, B, G, B, Y
R, B, G, Y, R, Y, Y, B, G, R, R, Y, R, G, B, Y, G, Y, G, G, Y, Y, B, R, R, B, G, B, G, Y, Y, R, R, R, B, B, R, G, B, B
Y, G, B, R, Y, R, Y, B, R, Y, R, B, G, Y, R, Y, R, G, R, Y, Y, G, G, G, Y, R, B, R, Y, B, R, G, Y, R, R, G, Y, R, B, Y
Y, B, G, B, B, R, G, R, R, B, G, R, G, R, R, B, Y, G, R, Y, Y, B, G, G, G, B, G, Y, G, Y, B, R, R, Y, R, B, Y, G, R, R
G, Y, Y, G, Y, Y, B, B, B, B, G, Y, B, R, R, G, G, G, B, R, R, R, Y, R, R, G, B, B, Y, R, G, R, G, G, Y, Y, R, B, R, B
G, R, Y, R, G, R, R, B, G, G, B, Y, B, G, B, B, Y, Y, G, B, G, R, G, G, B, R, Y, G, B, R, B, Y, B, B, R, Y, B, B, B, R
G, G, Y, B, B, Y, Y, R, G, Y, B, G, B, R, B, G, R, Y, B, B, Y, R, G, G, R, Y, Y, B, G, Y, B, Y, R, Y, B, G, R, G, R, G
B, Y, Y, R, Y, Y, G, B, Y, Y, R, Y, B, Y, B, R, B, R, Y, Y, Y, B, Y, B, Y, Y, G, B, G, R, Y, G, R, G, R, R, B, B, Y, G
```