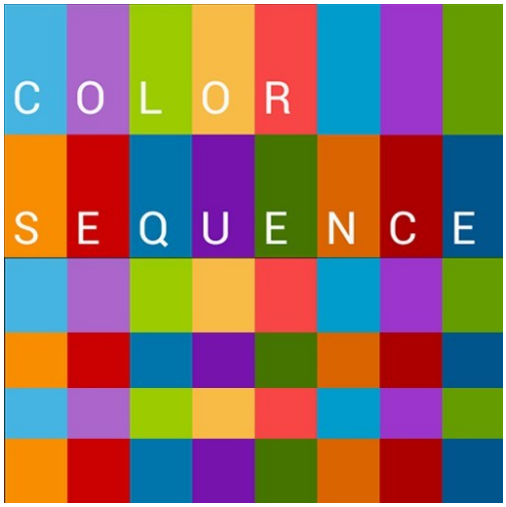
Longest Color Sequence assignment

Write a program that finds the ***longest adjacent sequence*** of colors in a matrix(2D grid). Colors are represented by ‘R’, ‘G’, ‘B’ characters (respectively Red, Green and Blue).

You will be provided with 4 individual test cases, which must also be included in your solution.



An example of your solution root directory should look like this:
solutionRootDir
| - (my solution files and folders)
| - tests/
| - test_1
| - test_2
| - test_3
| - test_4

1) Individual test case input format:

- First you should read two whitespace separated **32-bit integers** from the provided test case
- that represents the size (***rows*** and ***cols***) of the matrix.
- Next you should read ***rows*** number of **newline** separated lines of **8-bit characters**.

Your program should find and print the longest adjacent sequence (diagonals are not counted as adjacent fields), and **print** to the standard output the number.

NOTE: in case of several sequences with the same length – simply print their equal length.

Provided input	Expected output
3 3 R R B G G R R B G	2
4 4 R R R G G B R G R G G G G G B B	7
6 6 R R B B B B B R B B G B B G G B R B B B R B G B R B R B R B R B B B G B	22
1000 1000 1000 rows of 1000 R’s	1000000

2) Your program entry point should accept from one to four additional parameters.
Those parameters will indicate the names of the test cases that your program should run.

- Example 1: ./myprogram test_1 test_3
- Example 2: ./myprogram test_1 test_2 test_3 test_4
- you can assume that the input from the user will be correct (no validation is required)

3) You are free to choose a language for implementation between **C** and **Python**.

- Implementing the above description problem yields ***60*** out of ***100*** points.

4) If you provide a **C** solution:

- Your program should not cause **memory leaks**.
- You should also include a **Makefile** or use **CMake** tool to generate a **Makefile**.
- Running the GNU **make** command on your **Makefile** should produce a binary.
- **Bonus points**:
  - provide also a GNU **make clean** command if you are writing the **Makefile** OR
  - provide **out-of-source** build and build clean mechanism if you are using **CMake**
  - provide ***valgrind*** report on your solution with modes "--leak-check=full --track-origins=yes" (a simple screenshot is enough)
  - Implementing the above steps yields ***10*** out of ***100*** points.

5) Advanced (Master Jedi) section

- Use the **C** language for your solution
- Your solution should be build as a shared object (**.so** file)
- Your shared object should provide to it's users a public API function, which will resolve the "longest adjacent sequence" task.
  - Example: int32_t findLogestLen(/* some parameters */);
- Implement a separate **C**, **C++ or Python** solution, which will:
  - Load your previously generated **C** library
  - Parses each individual test case and provide it's data to the **C** library in order to find the solution;
- The usage of the separate **C**, **C++ or Python** solution will be the same as previously described
  - Example 1: ./myprogram test_1 test_3
  - Example 2: ./myprogram test_1 test_2 test_3 test_4
- Again – provide a **Makefile** or use the **CMake** tool to generate a **Makefile** for both the shared object(**.so**) and your solution(if you are using **C++**)
- Be creative when naming your binary and your shared object. Just keep in mind that we are very excited about the new Rammstein's single :)
- Implementing the above steps yields ***30*** out of ***100*** points.