COMSM1201 Resit

Flood It

Search for the game "Flood It" on the web, e.g. :
`unixpapa.com/floodit/?sz=6&nc=5` and learn how to play it (there are many `youtube` tutorials!)
This game is based on old paint-box tools, where you could flood-fill areas with colour. Each time you choose a new colour, the top-left region is flood-filled with this new colour. A region here, is defined as a connected set of cells with the same colour.

Basic Version (50%)

Write a text-based version , asking the user which 'colour' they wish to flood-fill from the top-left to look something like :

```
13213
23212
21232
32321
23321

Turn 1 : What number ? 2

23213
23212
21232
32321
23321

Turn 2 : What number ? 3

33213
33212
31232
32321
23321

Turn 3 : What number ? 2

22213
22212
```

```
21232
22321
23321
```

etc.
We will use `argc` and `argv` to allow the user a number of options in a fixed order when starting the game :

```
$ floodit
```

starts the game using 6 'colours' $(1 \ldots 6)$ and a 14×14 board.

```
$ floodit 5
```

starts the game using 6 'colours' $(1 \ldots 6)$ and a 5×5 board.

```
$ floodit 8 3
```

starts the game using 3 'colours' $(1 \ldots 3)$ and an 8×8 board.

```
$ floodit neill.txt
```

starts the game by reading a board from the text-file *neill.txt* and automatically inferring the board size and number of colours. This is useful for any automatic marking scripts. Submit your own file so I know the exact format you've used.

Notes

- The program should terminate when the board is finished i.e. one single colour fills the whole board. Your output should look as similar to the above as possible. Don't print out any other messages etc. unless there's an error.
- The maximum board size is 20×20 - you may use a static 2D array of this size, and when the required board is smaller, use only a portion of it.
- The maximum number of "colours" is nine $(1 \ldots 9)$.
- There is no maximum number of attempts allowed to "win".

Testing (40%)

Submit a `testing.txt` file that shows how your code was built bottom-up, through use of thorough testing. Show the testing you have used, how simple, small functions have been tested and combined together to give more complex functionality. Discuss bugs found, and self-reflection on the techniques used. Submitting a working program that was not informed by a testing strategy from the start will result in a poor (or failing) grade.

Extension (10%)

Submit a file `extension.txt` that describes an extension to the basic assignment you have undertaken. This could be to adapt the game rules, implement the game using SDL, or anything else that demonstrates some of the learning outcomes from the unit - check with me if your not sure.

Assessment

- Code in the assessment will be assessed based on the house coding-style, as usual. This means that getting the code to simply work may not be enough to pass - it must conform to the guidelines given in the unit.
- You can ask Neill at any time to see if your code would pass or not; I'll try and mark ASAP, but cannot guarantee this.
- You are welcome to submit early if you like - I'll mark your work as soon as possible. Submit via Blackboard and also by email, just in case.
- Do not base your code on any one else's - the normal rules for plagiarism apply here.