XJTLU | Computing

CPT108: Data Structures and Algorithms

Semester 2, 2023-24

Laboratory 2: Recursion - Extra Practice

1 Counting blob size

1.1 Problem Statement

Given a 2-dimensional grid of cells. Each cell is either filled or empty. Adjacent filled cells (either horizontally, vertically or diagonally) form a blob. There may be several disconnected blob on the grid. The problem is to count how many cells in a blob, given a cell (`x,y`) within the blob.

1.2 Your task

In the package you have downloaded from Learning Mall (LM), you will found a Java class:

`Grid.java`	The class which contains the basic functions that may be useful to count the number of cells in a blob.

The class takes a string as input where each row of the grid is separated by a newline character ('`\n`'), and will take any non-empty character as filled.

Your task is to complete the function `countBlobCells(int x, int y)` in Java class `Gird` (and other auxiliary functions, if necessary) which takes the `x` and `y` coordinates of a cell as input and output the number of cells in the blob; or `0` if the inputted location does not appear in a blob.

You will see the test program crash after executing the Gradle[1] wrapper `gradlew` from the terminal, since you haven't implemented the function(s) yet; only dummy functions are in the code you have been given.

```
GridTest > gridCountBlobTest() STANDARD_OUT
    maxRows=7
    maxColumns=12
        +------------+
        |012345678901|
    +---+------------+
    |000|   **       |
    |001| ***    *   |
    |002|     *      |
    |003|  ****    **|
    |004|  ****      |
    |005|  *    *    |
    |006|        *   |
    +---+------------+

GridTest > gridCountBlobTest() FAILED
    org.opentest4j.AssertionFailedError at GridTest.java:45
```

[1] Gradle: `http://gradle.org`

Details about these can be found in `GridTest.java` class inside the test folder, which also contain some test code that you can use to test your implementation.

Before start implementing your code, you should start analysing the problem by identifying:

- Input: What is given?
- Output: What is required?
- Constraints: Under what conditions?
- Abstraction: What information are essential?

ALL must be presented in some form of data that can be processed by computer.

For the *recursive step*, you should think about:

- What is the stopping case?
- What is the recursion step?
- How to ensure that the recursion step will eventually terminate?
- Are there any variables / methods that need to be added?
- What needs to be initiated on each computation?
- What needs to be initiated on each recursive call?