CS1083 Assignment # 3 – Fall 2021

Due: Wednesday October 6th before 11:00 pm (Atlantic) in the Desire2Learn submission folder. (See submission instructions below).

The purpose of this assignment is to practice two-dimensional arrays and loops.

This assignment is to be done individually. If you have questions, direct them to a tutor/assistant during a help session in the "Faculty of Computer Science Student Success Centre" team or to your course instructor.

Problem Description

> *"Underground salt formations are well suited to natural gas storage. Salt caverns allow very little of the injected natural gas to escape from storage unless specifically extracted. The walls of a salt cavern are strong and impervious to gas over the lifespan of the storage facility."*
> https://en.wikipedia.org/wiki/Natural_gas_storage

Consider a 2D model of an underground natural gas storage facility. In a 2D model, instead of the volume of a stored gas, we will be calculating the area of the cross-section of the cavern.

Assume the cross-section of the underground storage facility is modelled in a 2D array with all empty space (caverns) marked with zeroes (0) and the salt space (walls) marked with ones (1). Assume the gas can travel between adjacent array cells representing the empty spaces (caverns) only horizontally or vertically, and not diagonally. Also note that there may be multiple caverns underground but ONLY ONE has an opening to the ground level that can be used as an entry point (location of the valve). That is, assume that there is one zero in the top row of the array (row 0) marking the opening (the valve) to the cavern to which we have access.

Write a program that first reads the size of an array (rows, columns) representing the cross-section of the underground storage facility, then reads the 2D array representing the cross-section of the underground storage facility, and then calculates and prints the area of the cavern to which we have access (the one with the opening to the ground level). This area is equal to the count of array cells forming the cavern, including the cell representing the opening to the surface (the valve). To track or mark the spaces that have been counted, you

can change the 0's to another value (ie: 3). Assume that the data that is input to this program is valid (as defined in the previous paragraph).

An example data set is shown below (0's in the cavern are changed to 3's).

NOTE:

- For this assignment, you are NOT permitted to use recursion. You must solve this problem using loops.
- Multiple passes over the 2D array (i.e. multiple iterations of your loops) is fine.
- Refer to the additional instructions and requirements below and on the pages that follow.

Incremental Programming & Code Organization:

Be sure to use incremental programming and testing to make your job easier. For example, you might begin by first making sure that you can read in the input data and store it correctly. (You should not try to process the 2D array until you are first sure that you can build it correctly from the input data.) Next, you might check to make sure that you can find the location of the opening within your array. The idea is to build your solution piece by piece, while testing all along the way.

We also want you to think about the overall structure of your solution. You must think about how to decompose your solution into classes and methods. You should try to avoid long, complicated methods. If you find that a method is getting quite large/complex, this is usually a sign that you should consider restructuring/reorganizing your code, (perhaps by breaking that method down into 2 or more separate methods). Each method should serve a single purpose.

Sample Input

```
8 7
1 1 1 1 1 0 1
1 0 0 1 0 0 1
1 1 1 0 0 0 1
1 1 0 0 1 1 1
1 0 1 0 1 0 1
1 0 1 0 0 0 1
0 0 0 1 1 1 0
1 1 1 0 0 0 1
```

Sample Output

```
1 1 1 1 1 3 1
1 0 0 1 3 3 1
1 1 1 3 3 3 1
1 1 3 3 1 1 1
1 0 1 3 1 3 1
1 0 1 3 3 3 1
0 0 0 1 1 1 0
1 1 1 0 0 0 1

The area of the cavern is: 13
```

Testing Your Application:

In addition to all the testing you should do along the way as part of your incremental development, you should also test your application thoroughly after it is completed, using additional test cases. It is up to you to decide what test cases are required to prove to the marker that your application works correctly in all situations. Think about different scenarios when making up your test suite.

Before writing your solution (i.e. before writing any Java code), prepare a test suite (a file containing a series of maps) for testing an application that solves the problem described above. For each test case, be sure to indicate the input that you will use and the expected output. The test cases should include different cross section layouts and sizes. (Aside: Selected aspects of testing are covered in Section 7.9 in the textbook.)

Note:

- Your test suite should include the sample test case that is shown on the previous page plus at least 4 additional test cases.
- You may assume that the input is legal. That is: the first line of input includes the (correct) number of rows and number of columns for the array, the second line of input contains only one '0' (representing the location of the opening), and every other line contains only numbers 0 and 1.

When your Java classes are complete, run the tests from the test suite that you prepared in advance. If any tests fail, go back and modify your program to correct the error(s).

Commenting Your Code:

We've discussed the importance of properly commenting your source code, and the marker will be looking for comments in your solution.

You don't have to comment every line of code (that would be counter-productive and a bad idea), but we want to see javadoc comments in your solution. Include a comment for each class, for each instance variable and each method. Use @author, @param & @return tags where appropriate. Run the javadoc utility on your files and view the resulting html files in a browser to make sure that your javadoc comments were inserted/formatted correctly. You do not need to hand in the html files for marking.

Review your CS1073 notes about javadoc and/or consult Appendix I of the text.

Additional Coding Guidelines:

Reminder: **Readability matters!** Blocks within your code should always be properly indented. Use meaningful names for your classes, variables and methods. (Aside: Refer to Appendix F for additional tips regarding Java coding guidelines.)

Your electronic submission (submitted via Desire2Learn) will consist of two files:

i. a written report, with each part clearly identified with a section heading. Include:

   a. the source code for your solution (with javadoc comments included)

   b. the test report (test cases, explanation of test cases, test results)

This written report should be prepared using a word processor; we recommend using Microsoft Word (i.e. create a .docx file for your report). (Copy & paste your java source code & test cases (input/output) into the report document). Add appropriate headings for each part. Fix up the formatting where necessary, adjusting line breaks & page breaks to ensure that your document is easy to read. Use a monospaced font for your code to maintain proper indentation.) Once the report is complete and you've checked it all over, save the .docx file for your own records, and then **save a second copy in pdf format for submission**. (Note: Be sure to open that file in a pdf viewer to verify that the pdf was generated correctly.) The **SINGLE pdf file** containing your report will be submitted to the appropriate assignment submission folder Desire2Learn. (It is important that you submit a pdf file and NOT the original Word document. This pdf will allow the marker to write comments directly on your work to give you better feedback.)

Note: Please name this report as follows: YourName-A3-Report.pdf

ii. an archive file (.zip) that contains all of your work for this assignment. Make sure that your archive includes the complete source code (.java files – in case the marker wishes to compile & run your code and test cases). This archive should be submitted as a single file to the appropriate submission folder in Desire2Learn.

Note: Please name this archive file as follows: YourName-A3-Archive.zip