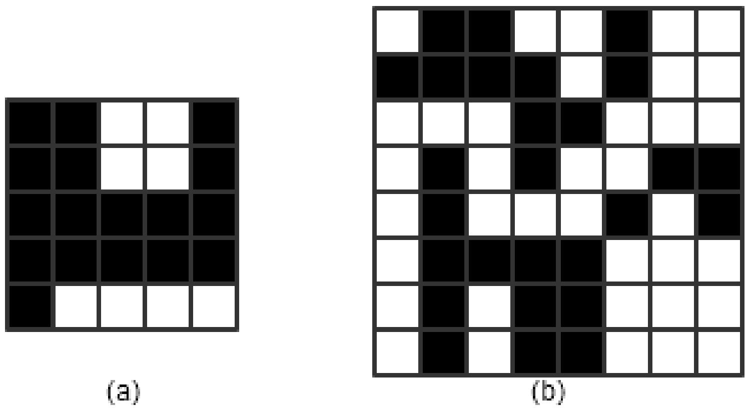
Mount Royal University | **Winter 2016**
COMP 2503 | **Instructor: Alan Fedoruk**
Programming III | **Feb 24, 2016**

Assignment 4 Recursion

Due Date: Monday March 7, before midnight
Weight: 5%
This assignment should be completed individually.

Description

Consider an $n \times n$ square of cells that are coloured either black or white. The problem is to determine the number of white areas and the number of cells that make up each area. For example, an 8×8 chessboard has 32 one-cell white areas.



Square (a) above has 2 areas both of size 4 cells. Square (b) has 5 areas of sizes 1, 3, 21, 10 and 2 cells.

Your task is to write a *recursive* program that inputs the configuration for a square then outputs the number of areas and the size of each area ordered from largest to smallest. When there are multiple areas of the same size you should indicate how many of each.

For (a) above the output would be:

```
There are 2 white areas
2 x 4
```

and for (b) the output would be:

```
There are 5 white areas
1 x 21
1 x 10
1 x 3
1 x 2
1 x 1
```

The finished program will be run like this:

```
cat test1.txt | java -cp A4.jar A4 > test1_out.txt
```

I will provide sample input and output files for the program. The output from your program must be *exactly* as shown in the examples.

Hint: Traverse the square row by and while you find an unvisited white cell, invoke a method to process a new area. In this method you will have four recursive calls. Once you have visited a cell, mark it as visited so you don't double count.

This is an example of the type of processing that can be done with image files. Instead of a simple 1 or 0 each cell could have an RGB colour and you could use this algorithm to find areas of similar colour.

Input File Format

The input file is a text file. It contains integers separated by whitespace.

The first integer is the size of the square, then the remaining integers are the contents of each cell in row-column order. A 0 indicates a black cell and a 1 indicates a white cell.

For the examples above, chessboard and (a) and (b) they would have an input files:

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

for (a):

```
5
0 0 1 1 0
0 0 1 1 0
0 0 0 0 0
0 0 0 0 0
0 1 1 1 1
```

and for (b)

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

You can assume that there are no errors in the input file.

Testing

A will supply some test input and output files for you to use. These files do not constitute extensive testing. You will need to do more testing to ensure all cases are covered.

I will be evaluating your program with test data that you have not yet seem and I will be trying to find flaws in your code!

What to Hand In

Hand in a single file, `A4.jar`.

Submit this to the Blackboard drop box provided. The jar should contain:

1. All of your `.class` and `.java` files.
2. A class called `A4`.

Grading

A detailed grading rubric will be provided.

I will run your program with the command line given above on three text files and compare your output to the specifications.

Encapsulation is important! Good clean design is important! Try to write elegant, simple code.

Outcomes

Once you have completed this assignment you will have:

- Implemented a recursive algorithm.