Implementing a Recursive Algorithm from a Description

Summary: In this homework, you will be implementing a maze generation algorithm from a plain English description.

1 Background

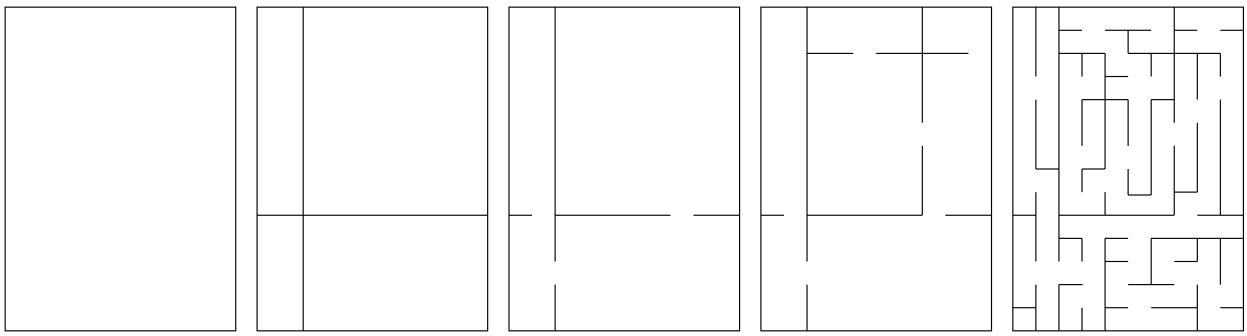
In this assignment, you will practice analyzing a text description of an algorithm, and then implementing it. There is no pseudocode, just text. In addition to this document, there is an additional PDF that reviews basic problem analysis in terms of recursion and this program. If you are having difficulty in completing this program, reading that document should help. Otherwise, you may skip the additional document.

This document is separated into three sections: Background, Requirements and Specifications, and Submission. You have almost finished reading the Background section already. In Requirements and Specifications, we will discuss what is expected of you in this homework. Lastly, Submission discusses how your source code should be submitted on BlackBoard.

2 Requirements and Specifications

Design and implement a recursive random maze generation algorithm. Start by downloading the attached source file. Study the starting code and make the following change: implement makeMazeRecursive(char[][]level, int startX, int startY, int endX, int endY) using the recursive division method. A general description is on Wikipedia: "*Mazes can be created with recursive division, an algorithm which works as follows: Begin with the maze's space with no walls. Call this a chamber. Divide the chamber with a randomly positioned wall (or multiple walls) where each wall contains a randomly positioned passage opening within it. Then recursively repeat the process on the sub-chambers until all chambers are minimum sized. This method results in mazes with long straight walls crossing their space, making it easier to see which areas to avoid.*

For example, in a rectangular maze, build at random points two walls that are perpendicular to each other. These two walls divide the large chamber into four smaller chambers separated by four walls. Choose three of the four walls at random, and open a one cell-wide hole at a random point in each of the three. Continue in this manner recursively, until every chamber has a width of one cell in either of the two directions." [40 points]



Notes:

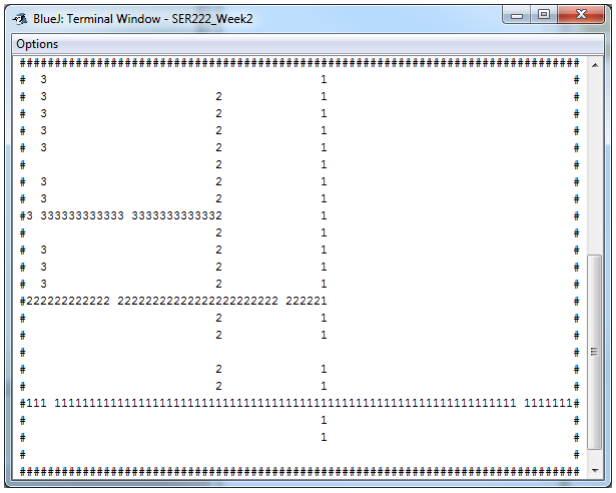
- This method is recursive. Do not call it on a sub-area when that area is less than 3x3 - this will cause walls to double up.

- Since this algorithm does not use any data structures or back tracking, it can create mazes that are not solvable.

Hints:

- (The screen shots below use numbers for walls in order to indicate which call to makeMazeRecursive created them - you should use only a # in your final submission.)
- It is not necessary to follow these hints, you may implement this algorithm as you see fit based on the description above.
- Consider creating a function called randBetween(int l, int u) that creates a random number in an interval - wouldn't this function be useful for deciding where walls should go?
- As a first step, implement the basic functionality to draw the two walls that divide the area into four rooms; don't proceed (i.e. add recursion) until it works. It should look something like:

- Then implement the recursive call for just one of the sub-areas; don't proceed until it works. Say, the top left. It should look something like:



- Then extend the code to the other sub-areas.

3 Submission

Source Code: Please rename the finalized code to "LastNameMazeGen.java" (e.g. "AcunaMazeGen.java"), and upload via the BlackBoard submission link. The class must be in the default package. Remember to change the class name as well. Do not include your project files.