CpSc 8380. Fall 2016
Programming Assignment 1: Mazes
Due Date: October 26; submit through link in Blackboard

Goal: To create an interactive GUI program written in Java 8, structured by the Model-View-Controller that will allow a user to dynamically create a maze either step-by-step or as a finished showing. The user will be able to select the "type" of rooms; the size from a limited number of sizes (you can choose wether its a dimension size or number of rooms) and the algorithm by which the corridors are created. The user should have the opportunity to see the same algorithm applied to all three room types at the same time. (You may choose the size.)

What you should submit:

Java - **SOURCE** only (Submit SRC folder maintaining packages, if used); No Jar files
PDF - Document presenting the details of your data structures;
All of your submission should be placed in a folder named for you (so that I don't have multiple Program 1 folders when I download to grade).

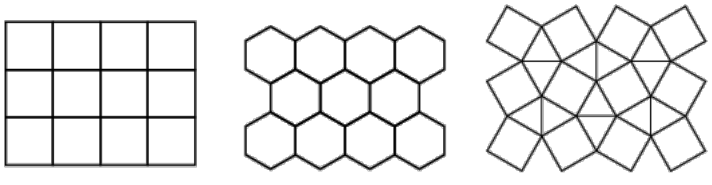
Additional General Requirements:

Each class should be contained in its own file;
Main method should be in a separate file;
Class definition should include brief overview of class;
Class definition file should include your name as author on a separate line from other documentation;
Class definition should include directory of "important" identifiers;
Identifier names should follow Java naming conventions (see tutorials) and should be of sufficient length to identify their purpose (no one letter names or one letter followed by number or unclear abbreviations);
Model, Controller and View should all include more than this single word;
Named constants should be used for constants which have specific meaning;
Extraneous documentation generated by IDE's should be removed;

Main method should instantiate the constructor and should "find" environmental considerations;
Controller is to instantiate both the model and the view;
View should encompass as much of the user's screen as the user's computer allows;

You are free to use any code provided in the tutorials at Oracle's Java site (however, you must document that use). You should take advantage of any data structures provided in the Collections Framework or Algorithms provided with this framework. Be advised that if you seek other code from any other source, it will reduce your grade. (Of course, it must be well documented that you are using such code and it must be "in your own words" so to speak.)

This program is to create a program which will will be driven by choices made by the user to generate a random maze. The user will select in some order the underlying shape of the rooms; the size of the maze (excluding walls); and the algorithm used to create the maze. The underlying shape of the rooms is to be chosen from squares,



hexagons and the semi regular tessellation composed of squares and triangles seen to the right. Your goal in writing this program is to use or create suitable data structures to accomplish the model. Creating a “maze framework” would certainly add to the value of your project. Drawings from Wolfram Mathworld.

Your maze should have one designate entrance and one designated exit (both on the outside face) and without loops (no islands) and is connected. Such a maze is often designated as “perfect”. A perfect maze may have dead ends and it is expected that the mazes that you generate will have such dead ends. The paths in the maze can be described by a graph (as in Graph Theory). A perfect maze will be described by a tree.

Your program should incorporate the following algorithms in the creation of such a tree. Two of the algorithms are standards for finding a minimum spanning tree in a graph. Of course, we do not have weights on the edges of the graph underlying the tessellation, so we will simply use them to find “spanning trees”. Each can be expected to create dead ends in the maze making the maze to be “more entertaining” to try to solve. These two algorithms are Prim’s algorithm and Kruskal’s algorithm. The third algorithm I believe was created solely to generate a maze (of fixed width, but possibly infinite length) that is perfect. It is known as Eller’s algorithm and finishes a “row” so to speak in the maze before moving on to the next “maze”. There is randomness in the creation process that allows different mazes to be created.