CSc 225 Assignment 6: Graphs

Due date:

The submission deadline is 11:55pm on Monday, July 27th, 2020

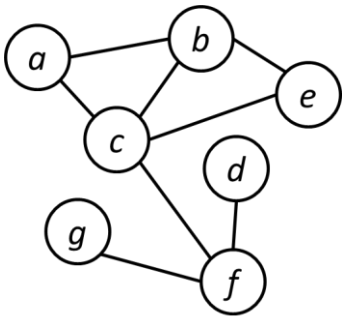
How to hand it in:

Submit a **.pdf** file (for Part I - questions 1, 2, and 3) and the **`GraphAlgorithms.java`** file (for Part II - question 4) through the Assignment 6 link on the CSC225 ConneX page.

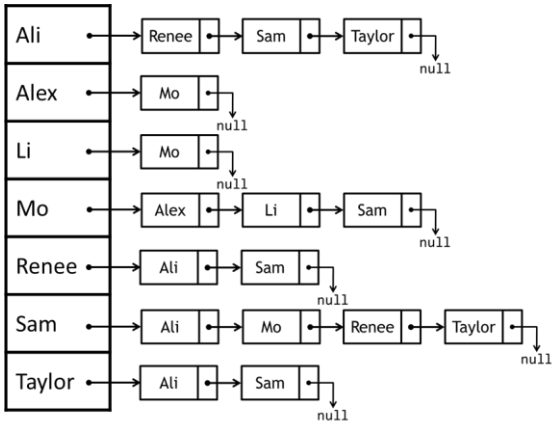
Part 1: Graph Theory and Representation

1. Consider the following graphs presented in three different representations:
   - I. a drawing of vertices and edges
   - II. an adjacency list
   - III. an adjacency matrix

Graph I:



Graph II:



Graph III:

	1	2	3	4	5
1	0	1	1	0	1
2	1	0	0	1	0
3	1	0	0	0	1
4	0	1	0	0	1
5	1	0	1	1	0

Answer the questions on the next page about the three graphs.

a) Draw graphs II and III (a drawing of vertices and edges similar to Graph I)

b) Convert graphs I and III into adjacency lists (similar to Graph II)

c) Convert graphs I and II into an adjacency matrix (similar to Graph III)

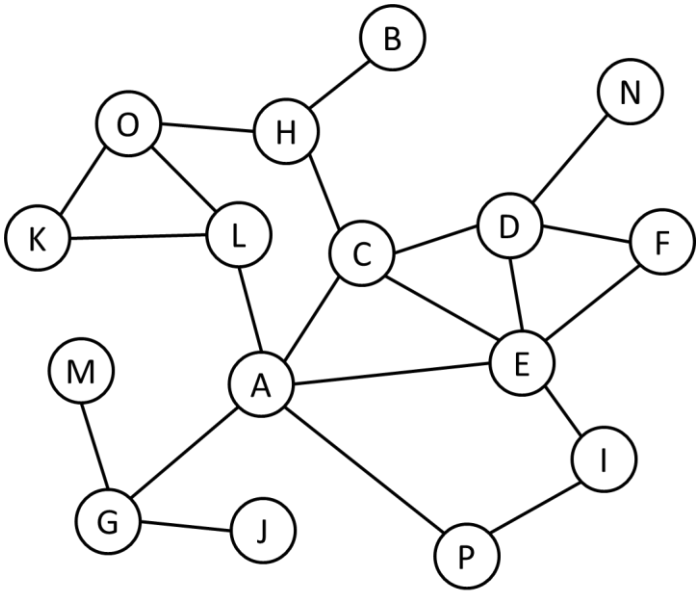
d) What pairs of graphs are **isomorphic?** Indicate the mapping from one graph's vertex labels to the other for each pair.

e) Draw a valid spanning tree for Graph II rooted by Sam.

f) Provide an associative array Parents for all nodes in the spanning tree of Graph II rooted by Sam. Parents should consist of a set of key-value pairs where each vertex in the graph has a key, and the associated value for each key is the node's parent in the spanning tree.

2. Provide the order of vertices visited for both a pre-order and post-order traversal of the following graph. Begin each traversal at vertex A.

   When choosing between neighbours to visit first, assume the algorithm chooses the neighbour that comes first alphabetically.



Pre-Order:

Post-Order:

3. After social distancing ends, a group of 6 students decide to have a LAN party to study together for their CSC final exam. Each of the 6 students brings their own laptop.

   Each laptop is connected to 0 or more of the other 5 laptops in the room (a laptop cannot be connected to itself). Prove that there are at least two laptops in the room that are directly connected to the same number of other laptops.

   (Hint: try solving this problem assuming each laptop is connected to at least one other laptop first. If you can solve it that way, it may be helpful in solving the problem stated as it is above)

Part 2: Programming with Graphs

4. Download the a6.zip file containing all of the starter files for this programing component of this assignment. The file can be found in the Resources > Assignments > a6 section on ConneX.

   The `A6Tester.java` file has been set up to allow you to visualize the results of your DFS and BFS flood fill (colouring) algorithms.

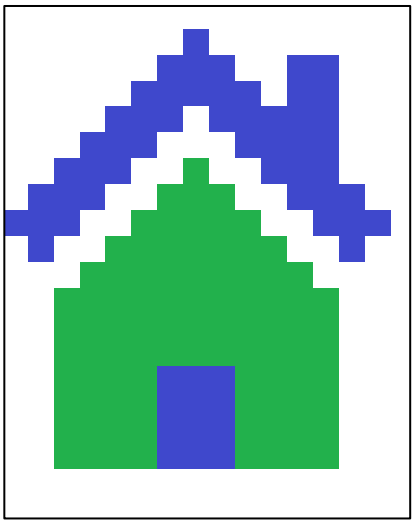
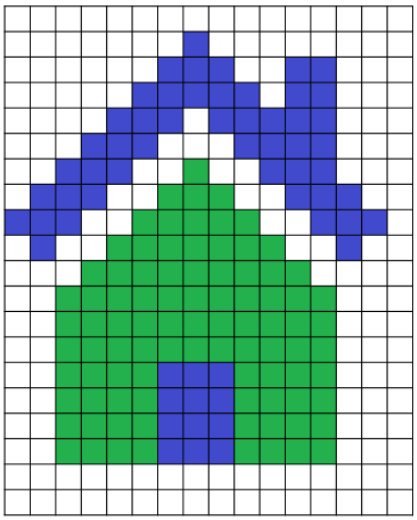
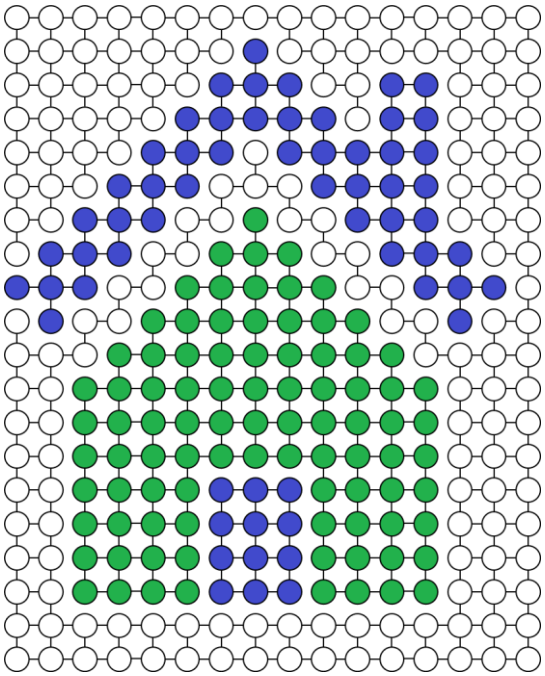
   Description:

   As demonstrated during lecture, images can be represented as 2-dimensions arrays of pixels. Usually, each pixel's colour is based on Red, Green, and Blue (RGB) components between 0 and 255. An RGB value of (255, 0, 0) would be red, whereas (0, 255, 0) is green, and (0, 0, 255) is blue.

   Many image processing tasks use information from neighbouring pixels. For this assignment, you will be representing images and graphs, where each vertices neighbours are adjacent pixels in the corresponding image of the same colour.

   For example, look at the image below on the left depicting a house (with a blue door, blue roof and chimney, and green frame). The middle image below depicts the image as a grid of pixels.

   The image on the right shows the same image represented as a graph, with the following properties:
   - each vertex in the graph corresponds to a pixel in the image
   - each vertex is connected to adjacent pixels of the SAME colour

   Your task will be to implement the flood fill operation found in many image editors. To perform a flood fill, one chooses a colour and clicks a point in the image. All pixels connected to the point clicked of the same colour are then changed to the new colour.

   This same process will be done using DFS and BFS traversals. When a point in the image is clicked, all connected vertices will be changed to the same colour.

The only file you need to add code to is `GraphAlgorithms.java`.

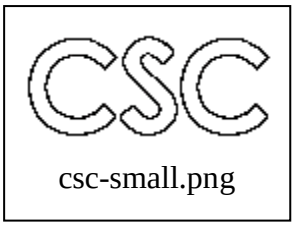
It will likely be necessary to read through the `PixelVertex.java` and `PixelGraph.java` files to understand how the graph has been set up, as well as the operations available to you for each vertex (PixelVertex) in the graph.

You are not expected to understand the code in the `A6Tester.java` file, but for those of you who are curious, feel free to experiment with it. Your solution must work with the current version of the `A6Tester.java` file. `A6Tester.java` is executed with the name of an image. For example:

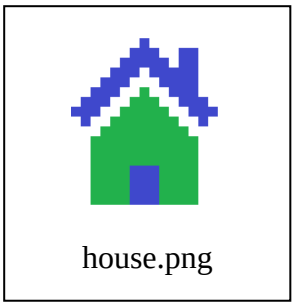
```
java A6Tester house.png
```

opens up the testing visualization program with the house image shown on the previous page.

There are four other sample images provided for you. All of them are very small, so if you want to animate the flood fills, you are able to without waiting too long.

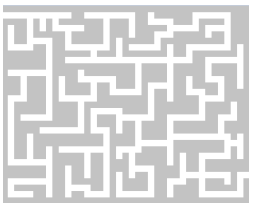


csc-small.png


house.png


smile.png


lines.png


maze.png

Feel free to make additional images and post them on the forums. Good luck!