CSCI 1311 Final Exam Review

The final exam will be formatted very similarly to the midterm. There will be 10 normal questions plus an extra credit question. The questions will be roughly one third short answer, one third coding, and one third tracing. You can get partial credit, but only if you show work/write something down. You will be allowed to use both the Greenfoot API and the full Java API. You will not have access to on-line tutorials, class notes, in class code, or other resources.

The exam will explicitly cover the second half of the semester. Concepts from the first half of the semester will appear only implicitly. That is to say that the topics from the first half of the semester are generally foundational to those in the second half of the semester. So while the questions will be asked about concepts from the second half, they will wind up requiring you to use information from the first half. The topics that were covered in the second half of the semester are listed below along with some general indications on what you should know about them. I don't claim that this covers absolutely everything that will be on the exam, but it gives you a good indication of what I plan to put on the exam.

Files – Know how to read from and write to files. You want to be able to write code that does this as well as read code that does this. It doesn't have to be CSV, it can be any fairly simple text information.

Spreadsheets – Understand the basic concepts of how we did "spreadsheet" processing. You should be able to explain, code, or trace the processing of a table of data using loops and basic logic.

HashMaps – You need to understand what these data structures do and be able to write code or trace code involving them.

Optimization – Know what optimization problems are and how we use computers to solve them. We have looked at two basic approaches to this that are covered in the topics below.

Graphs – You should know what these are and be able to communicate in words or code, how we work with them.

Minimum Spanning Tree – Understand what the minimum spanning tree problem is in terms of graphs. Also understand how the code that we wrote to solve this problem works. Know it is greedy, what that means, and how it qualifies as that type of algorithm. You will not have to be able to write a minimum spanning tree solution in code.

Recursion – Know what recursion is and what it is used for. You should also be able to write or trace recursive methods.

Shortest Path – This was another optimization problem that we did on a graph. This one is not greedy, it required using recursion to check all paths. Be able to explain how it works and trace similar code.

Scheduling – This was another optimization problem that we did using recursion. Understand how our code for this worked. You should be able to explain it and how you would modify it for different purposes. You will not have to write this from scratch.

Applets – Know how to program applets. This includes the five main methods that are part of the functioning of an applet as well as the use of listeners/events for getting user input and Timers for creating animation.