Final Project: Graphs

Specifications: Create a working undirected, unweighted graph class with all the standard methods for a data structure of that type.

Instructions: This assignment is your opportunity to demonstrate everything you have learned in this class. There are not any detailed specifications because this assignment is your demonstration of your understanding all of the following concepts:

- Loose coupling.
- Separation of interface from implementation.
- Developer testing.
- Best practices in procedural and object oriented programming.
- Understanding of algorithms.
- Understanding of graphs.
- Your graph has to be capable of growing to any size and any configuration for an undirected, unweighted graph and must have the concept of an “id” and “data” per node in the graph. It’s up to you what “id” and “data” means.
- Use this invite to create an empty repo (it just has a README.md) https://classroom.github.com/a/YRydNfHE
- Submit your final project normal as any assignment by submitting your link.

Analogy: Imagine you spend all semester in a “research writing” class. You would have been taught writing techniques and best practices, how to cite information, how to write a table of contents, how to format and structure your writing, etc. Each assignment in the semester would have walked you through each individual concept with strict guidance to learn each specific technique. Then at the end of the semester, your final project is simply “write a research paper.” At that point you should know all the writing techniques, processes, and best practices to write a complete research paper on your own. You should be able to research a topic and complete a proper research term paper on any topic.

This assignment is similar in concept. You’ve learned all the proper techniques for programming at this level, so now you should be able to get a simple instruction such as “make this-or-that ADT” and it’s up to you to put it together properly. It’s up to you to do the research, study sample code, learn the algorithms, and design and build a graph class demonstrating you understand both graphs and all the best practices presented in the semester.

Grading: You will be graded on all best practices presented in class, architecture, and proper developer testing. To achieve 100%, your architecture and best practices must be close to perfect and you must have all **basic** graph functionality complete. It’s up to you to intelligently decide what “basic” functionality is based on what you learned in class throughout the semester (if in doubt, ask). Partial credit will be given based on how far you get and how well you structured your class, loose coupling, and how well you performed testing. You will also be graded on your repo and testing with the following guidelines:

- Failure to commit often, small, and smart will result in an **automatic -10% penalty** regardless of your code quality.

- Any stray files in your repo will result in an **automatic -10% penalty** regardless of your code quality.
- If you do not have proper comment headers and/or do not use the Write Submission feature (not the comment section), and/or do not submit your link correctly, it's an **automatic -10%** penalty regardless of your code quality.
- Failure to use the correct branch in your repo (main) will result in an **automatic -5% penalty** regardless of your code quality.
- **Assignments that do not compile for any reason will not receive a grade above 50%**

Submission: When you are ready for grading, use the write submission feature in Blackboard and submit your repo link (this is the URL in the browser, not your SSH or .git link). If you need to fix/change something after you submit but *before* it is graded, just fix/change it and push again.