CSCI 2270 – Data Structures and Algorithms
Instructor: Hoenigman
Final Project
Due Dates:
Phase I: April 19, 5pm.
Phase 2: April 26, 5pm.
Phase 3: May 3, 5pm.

In this final project, you will be creating a code library of your own design and storing your code on gitHub. Additionally, you will be contributing to and evaluating other projects by students in your class.

The project is divided into three phases.

Phase I
In recitation, you will learn about gitHub, create a gitHub account, and the code repository for your project. You will also contribute a Read Me file to your code repo that describes what your project is.

Phase 2
Generate the code for your project and add it to your gitHub repository. The majority of the coding you will do for the project occurs in Phase 2.

Phase 3
Evaluate three other projects by your classmates using a grading rubric that will be provided. Also in Phase 3, you will contribute to another classmate's project, either by adding a feature to their code or fixing bugs in their code.

Phase 1 Requirements:

Walk through the steps in the recitation exercise to create a gitHub account and inform your TA of your gitHub username. If you already have a gitHub account, you are welcome to use that username.

Create a Read Me file that describes your project. For Phase 1, your Read Me should have at least one paragraph that contains the project summary and the other section heads. Fill out as much of the Read Me information as you have available in Phase 1, it will save you the work of having to do it later. Your final Read Me file, once your project is complete, will include all the instructions needed for someone to download your code and run it on their machine.

Submit the URL for your project repository, and the user ids for members of your project team, to the Moodle quiz Final Project - Phase Submission. If you are working alone, enter "None" for project teammates on the Moodle quiz. The project repo should be created under one person's account, but everyone on the team should submit the URL for the project.

Read Me Requirements:
Your Read Me file needs to contain the following sections. Not all sections need to be written in Phase 1.

Project Summary
One paragraph description of how the program works and what it does.

How to Run
This section contains instructions, with examples, of how to run your program. You should also include a link here to the project documentation that you will provide in a separate file.

Dependencies
This section contains a description of any dependencies that your program requires. For example, if your program relies on another third-party library that needs to be installed, you should provide a description of where to find that library and instructions for how to install it.

System Requirements
Is your program for Windows, Mac, Linux? Are there additional system requirements other than the operating system?

Group Members
List the people who worked on the project.

Contributors
List the people who were not on the project team, but may have contributed comments, enhancements, and bug fixes.

Open issues/bugs
List any known bugs in the project, and any open enhancement requests.

Phase 2 Requirements:

In Phase 2, you will be writing the code for your project. There is no set assignment that you need to implement, it's all up to you. Do whatever interests you and satisfies the following technical, design, and documentation requirements.

Phase 2 technical requirements:

- Contains at least one class.
- Contains at least one data structure that we've studied this semester.
- Your class should contain at least 10 public methods.
- Contains a well-documented driver file to run your library code.
- Your library can't be an assignment from the semester with no additions. However, you can start from an assignment and extend its capabilities.

- e.g. A Graph class to calculate Dijkstra's algorithm not enough. A graph class with several shortest path algorithms is.

Phase 2 code design requirements:
In Phase 2, your code should be designed to match these stylistic requirements: Your class interface and implementation should be in separate files. This is the style we've been using all semester with Graph.cpp and Graph.h being separate files, for example. Your driver file is a separate source file that uses the public methods in your class. In our assignments, an example of a driver file is Assignment8.cpp. If you have other helper functions, they should be stored in a separate source file and the function prototypes should be listed in a .h header file. For example, if you create a file called utilities.cpp, there should also be a utilities.h that is included in any .cpp files that use the functions in utilities.

Phase 2 documentation requirements:
Your project repository needs to include a documentation file that describes the public methods available in the library. A user of the code should be able to read through the documentation and know, for each of the methods, the expected inputs and their types, and the expected outputs and their types.

Your documentation needs to describe the pre- and post-conditions for each method. We will talk more about this in lecture.

Phase 2 collaboration recommendation:
If you are working in a team, it is recommended that each teammate be responsible for a separate file. Merging file changes is difficult in gitHub and it's very easy to lose work by overwriting one person's changes.

Phase 3 Requirements:
In Phase 3, you are going to evaluate and contribute to other projects. This is the collaboration phase of the project.

Phase 3 evaluation requirements
Select three other projects to evaluate. Your TA will be monitoring which projects everyone selects to verify that we get good coverage over all projects. If you would like to evaluate more than three projects, you are welcome to do so. (If some projects are not selected for evaluation, we may need to assign some projects for evaluation. You will be provided with an evaluation rubric that you will use to assign a score to each project on a scale of 1 to 10.

The primary criterion for evaluation is whether you can run the project code by stepping through the instructions in the project Read Me.

Phase 3 contribution requirements

The Phase 3 collaboration phase also includes an opportunity for you to contribute to another project. This contribution can be in the form of adding a feature to another project, or resolving a bug in another project. All project repositories will have an Issues section that lists the bugs and feature requests for that project. You will learn more about Issues in the gitHub tutorial in recitation. To contribution to a project, you can clone the code to a local directory and then issue a pull request to have your change added to the project. The project owner will need to be responsive to pull requests to have your change included. Therefore, there are two components to this collaboration requirement: changes that you make to other classmates' code and responding to changes that your classmates make to your code.

Other Project Details

Grading

Each of the phases will contribute to 1/3 of the final project grade.

Teams

Yes, you can work on teams. Teams of two students are the recommended size. If you have a team larger than that, you are likely going to have a coordination nightmare. The project repository will only be under one teammate's account, but we will be able to track each teammate's contribution to the project. Each teammate is responsible for evaluating and three other projects and contributing to another project. For teams larger than two students, we will be expecting to see more-advanced functionality in your project. Check with your TA to verify that your project has an acceptable level of complexity.

Project ideas

- It can take some thought to come up with a project idea. Here are a few suggestions:
- Extend an assignment from the semester to include additional features.
- Implement a Python library that doesn't exist in C++, or at least a portion of the library. (Be brave, learn how to do cURL requests.)
- Implement a data-specific data structure. For example, you could implement graph traversal algorithms that solve a specific type of shortest path problem.