CS 133, Spring 2022
Final Programming Project (60 points)

Due in six milestones on 5/9, 5/16, 5/31, 6/8, 6/10 and 6/11

For this assignment you will work in a group of 2-3 students to implement a program of your choice. Your program will be implemented in C++, using what we have learned throughout the C++ series. This is meant to be an opportunity to create something that you find interesting and exciting. **Your program must meet all of the requirements below**. However, you are welcome to add as much additional behavior as you like.

Concept:

This project is very open ended as it is meant to be something that is meaningful to you. Although it is very open ended, it will be important to make sure that you meet all the requirements listed below and that your project is the required size and complexity.

Implementation Requirements:

1. *Your program must be implemented in an object-oriented style with at least 5 separate classes*.

   The interface should be separate from the object that manages the core logic. If you need to group data together it should be its own type.

2. *The program must provide some method of user interaction.*

   This can be through the console, through a GUI, through differing input files or some combination of the three.

3. *The program must include some manipulation of a linked structure*.

   This can be a list, tree or graph. Whichever you choose, all node manipulation should be done in one class.

4. *The program must use at least two of the following standard library structures:* `vector`, `set`, `unordered_set`, `map`, `unorderd_map`, `stack`, `queue`.

5. *The project must be similar in size to your final project in CS 132.*

Project Ideas:

Your program can do anything that your group likes as long as it is school appropriate. Here are some ideas (although **you do not have to use any of them**):

1. A class scheduler, similar to CtcLink. This would allow the user to add and remove classes but it would need to check that no classes were added that were the same date and time as a previous class. It would also need to check that classes had spaces left and that students had the right prerequisites before allowing them to add a class
2. A game where the user can play either against another person or against a computer AI. This cannot be too similar to projects 2 or 4.
3. Categorizing, identifying or generating documents based on the counts and orders of words. You could use counts of n-grams to identify authorship, generate documents that appeared to be written by someone by using the most common n-grams in their writing. You could also group documents into categories by topic, likelihood to be spam email or likelihood to be fake news.
4. A recommender system like Netflix's. Read in data from a file about current users' movie ratings. Then when asked to generate recommendations for a particular user compute the similarity between the user's ratings and other users ratings and recommend movies the most similar user has also rated highly.
5. A program that allows the user to ask for the quickest path between locations on a map. The path must stick to streets. The streets could have traffic ratings and/or elevation ratings. The program could allow the user to input new information like which roads are currently closed.

Development Strategy:

Start by agreeing with your partner(s) about what your program should do. Next, divide your desired behavior into categories. What is the minimum viable product (the simplest, most bare-bones version that will meet the requirements)?

Next, what additional behavior or features would you like to add? Prioritize these in order of importance. This will help keep you on track and meeting all requirements for full credit even if your original goal is very ambitious.

Next, figure out what classes and data structures you will need and which classes the data structures will be part of.

Next, implement the classes in your inheritance hierarchy. These will be somewhat similar to Critters from project 5.

It will be very important to test your code **incrementally**. You will want to write several small main programs as you develop your classes to make sure that they are working correctly.

Style Guidelines:

This is a large assignment that encompasses almost everything that we have covered in class this quarter. Keep all previous style guidelines in mind as they will apply here as well.

In particular, make sure you follow proper **object-oriented** programming style. You should **encapsulate** the data inside your objects, and you should not declare unnecessary data members to store information that isn't vital to the state of the object. Use **inheritance** and **template**s when helpful to eliminate redundancy.

As always, follow past guidelines about proper indentation, spacing, identifiers, and localizing variables. Make sure to use good Boolean Zen, and factor your conditionals.

Place comments at the beginning of each header file including your group information and documenting that header's behavior, on top of each member function header, and on any complex code. Remember that your main public comments should be in your header files not in your cpp files. However, your cpp files should contain comments on the file header as well as internal comments in member functions explaining complex sections of code.

Make sure that you remove any debugging code before submission.

Group Work Guidelines:

You will work with one or two other students to complete this project. Group work can either be way better than working alone if your group works well together or way worse if it doesn't. In order to ensure that everyone has a good experience you will be required to complete and submit documentation of the following activities to receive full credit:

1. Spend **at least 6 hours** coding together. This means writing or debugging code while looking at the same screen. It includes discussing the best way to implement a function or feature and actually writing the code together. **The more time you spend coding together the more you will get out of this project.** The groups happiest with their experience with the CS 132 final project were the groups that coded together most of the time.
2. If your group is going to spend time coding on your own this should be done by all group members not just one. This means that there shouldn't be a situation where the group members code two classes together and then one group member writes four other classes on their own. If there are four other really complex classes to write and you need to complete them separately these should be split evenly between the group members. Split tasks based on the amount of work not number of classes as some will be much harder than others.

   Although tasks should be split equally, it is just fine if group members end up collaborating more on one class versus another to help debug, alter the interface or otherwise perfect the code. The goal is just that everyone is contributing.
3. **Check in regularly** (at least every 2 days) with your other group members about your project progress.
4. Every group member must keep a log of the following:
   a. The time they spend working on the project.
   b. The classes/functions that they are the main author of.
   c. The additional classes/functions that they helped debug.

   You will have the option of using a GitHub repository to maintain your code for this project if you would like. However, this is entirely optional. If you choose to do this you may find it simplest to put the above information in your commit messages. That way, when you are ready to submit, you have it all in one convenient place.
5. **Let Allison know right away if your group is having trouble meeting these guidelines**.

Your group's project will receive a percentage grade as specified later in this document. By default, this **grade will be shared among all group members**. However, if particular group members contribute a large amount more or a large amount less than their partners, they may be subject to an individual multiplier between 0.5 and 2. Refusing to meet, failing to communicate, not doing one's share of work, or other negligence may result in a multiplier less than 1.0. This will only be done in circumstances where the group members and instructor agree clearly that there was a significant lack of work performed. If one of your partners is not doing his/her share of the work, it is your responsibility to attempt to convince them to contribute. If this is not successful, notify Allison promptly. If we are not notified of a group problem until the last minute, there is not a good chance that a group problem can be resolved, nor different grades assigned.

Deliverables and Deadlines:

You will submit your work incrementally over the rest of the quarter. The work is divided into the following "**milestone**" deliverables. Each milestone can be submitted **up to 24 hours late for a -5% penalty, and will not be accepted more than 24 hours late.** Each phase is graded on **functionality ("external correctness") only**, aside from Milestone 4. If any group members have late days remaining, those group members will lose late days rather than points.

Milestone 0: Project Partner Request Due Monday, May 9, 11:59 PM (1% of grade)

Fill out the partner preference survey linked on the course website.

Milestone 1: Project Planning Due Monday, May 16, 11:59 PM (5% of grade)

A document containing the following information:

- Your group members names.
- A paragraph describing what you will be simulating
- A list of all of the classes you plan to implement and for each class all of their public member functions
- A list of the data structures you plan to use, what you plan to use them for and which class they will be in
- An explanation of how and where you will use a linked node structure.

This should be submitted either as a `.txt` file or as a PDF.

Milestone 2: Beta Code Due Tuesday, May 31, 11:59 PM (24% of grade)

The first version of your code implementation will be called the "milestone" version. For this version we want to see that your group has completed a significant chunk of the work of the project, though we do not expect the program to be fully functional. We expect that by this date, we should be able to compile and run a version of your program successfully. When the program is run, we should be able to at least see some text output displaying the results of the core complex calculations in your project. It is allowed for this milestone to contain some bugs as long as the general functionality can be run and tested.

Milestone 3: Project Presentation Due Wednesday, June 8, 2:50 PM, in class (5% of grade)

On the last day of lecture in class, we will perform short demos of all groups' projects. During these demos, one of your group members will run your simulation on his or her laptop while your group speaks briefly about your work and simulation and the features you chose to implement. **All members of your group must be present in lecture on the last day** of class and speak briefly about your project to receive credit for this milestone. Your project should be generally finished by this date so that it can be demoed successfully. But your source code does not have to be 100% complete for submission until the Friday of the last week of classes as documented below.

Milestone 4: Gallery GIF and Description Due Friday, June 10, 11:59 PM (3% of grade)

You will submit an animated image of your project running as well as a paragraph describing what your project does. These will be posted on the course website on a final project gallery page.

See separate linked instructions for how to create an animated image (GIF) of your program.

Milestone 5: Final Code Due Friday, June 10, 11:59 PM (60% of grade)

You will submit the final version of your code and resources on the last day of classes for spring quarter. The final version will be graded more strictly than the other milestones and will be expected to meet all requirements for the project as described in this document. The correctness of your code will be graded by running your program and manipulating its

interface or input files. It will be very important to **include good documentation about how to run your project**. Exceptions and segfaults should not occur under normal usage. Your code should not produce any debug output.

Milestone 6: Group Evaluations Due Saturday, June 11, 11:59 PM (2% of grade)

After you've submitted your final code, you will also fill out a short survey about each of your project partners and what work you and the others did on the project. This will help us make sure that every group member contributed significantly to the project.