CS 172 Final Programming Project

GRADE:

CATEGORY	POINTS
Project Proposal and Requirements Specification	15
Code Implementation	60
Presentation	15
Individual Summary	10
TOTAL	100

DUE DATES

- **May 10: Project Proposal and Requirements Specification (1 page at least).** Essentially the design for your project – IMPORTANT!
    - Summarize the problem, how you will begin to approach the problem and anticipated challenges.
    - A clear problem definition. State the requirements – what is required of the system? What must it accomplish or provide? Do you make any assumptions?
    - The design of your project. For instance, what classes, what properties, what behaviors, etc. will be required? Include a **UML class diagram** complete with designated member methods and variables.
    - Put names of the team members (No more than two)
- **May 17 for CS172-2 and May 19 for CS172-1: Presentation**
- **May 20: Final Project Submission**
    - Updates to specification – did you change the design along the way? Did you add/remove features? Make sure your document from May 10 is up to date.
    - WhitGit commit comments
    - Application code

PROJECT DELIVERABLES

- In your FINAL_PROJECT WhitGit folder, you should have the following items:
    - **project_proposal_and_requirements.docx (1 page at least)**
    - **project/ -** a folder that contains all the source code for your final project.
- Email me the following document indivisually
    - **individual_summary.docx** – summarize the work you did **(1 page at least)**
        - What did you use from class to design and implement your work?
        - What concept(s) from class were made more
        - What did you need to learn outside of class?
        - What surprised you most about this work?
        - What would you do differently?
        - What would you do in "version 2" of your project?

REQUIRED FOR EVERY PROJECT

- You **MUST** use **object-oriented** programming.
- You **MUST** have separate class definition header, and class implementation files
- **Well commented code**.
- As you develop your code, use good software engineering practices.
    - For example, define the **clear requirements** at the beginning to help you understand what needs to be implemented, AND design **UML class diagrams <u>BEFORE</u> beginning to code**.
    - When you commit a change to WhitGit, your commit comments must be meaningful
- Employ the concepts we have learned this semester as appropriate: templates, vectors, pointers, dynamic memory, File Input/Output, perhaps even inheritance and polymorphism. This is not required but it will help you get better in the craft. This is your opportunity to experiment and see what you can do!!
- Consider reaching beyond into the latter chapters in the textbook and exploring other more advanced data structures of the STL Library (e.g. linked lists, stacks, queues, hash tables/dictionaries, binary trees, etc.)

EXTRA CREDIT

You can pick up some extra points on this final project by including concepts discussed after the exam:
Inheritance: +3%
Polymorphism: +5%
Recursion: +3%

For inheritance or polymorphism, you must show your work in UML as well as take advantage of the concept in a useful way. Recursion is to be shown in your code, in place of a loop.

GRADING CRITERIA

20%	**Problem analysis**: Does the design exhibit a general understanding of the problem definition?
50%	**Execution and Results**: Does the project **work** correctly and generate the correct answer(s) according to the problem specifications?
25%	**SW Development Quality**: Is the project well designed, contains comments and contains clean, well-organized code. Does the project utilize appropriate and effective structures and programming concepts? Is the code readable and easy to follow?
5%	**Efficiency and Creativity:** Top project designs show ingenuity, creativity, and efficiency.

SOME FINAL PROJECT IDEAS

The following are some ideas for final projects. **You are not limited to these ideas.** Please be creative! Your problem should be of equal or greater complexity than these suggested problems. You can also look through advanced problems in your text or other textbooks for more ideas.

- **Text-based model/simulation** – Implement a program that simulates a real-life scenario. Examples are simulating an airport, air traffic scheduling, processes of a hospital, a manufacturing floor, elevators in a skyscraper, a physics problem, a biological model, a societal or anthropological issue, scientific problems.

- **Interactive tutoring project** – In CS-171 you implemented an interactive tutoring program for students. Redesign this using Classes/Objects and other advanced concepts you learned this semester. Perhaps you will like to keep track of student's progress/scores in files, read and present questions from different topics from files, etc.

- A **Business application** - For instance, a system might use a number of files and calculate profits (as per purchases, sales, expenses), might calculate payroll, keep track of bank accounts, etc.

- A **Bioinformatics application** - Develop an application for analyzing genetic sequence information (e.g. search a sequence for potential genes), or, develop a rudimentary application for protein structure prediction based on the amino acid sequence in a protein. Another option would be to develop software that can create phylogenetic trees based on DNA/RNA information.

- **Explore** implementing a software solution to an advanced problem in your other classes, i.e. math, chemistry, physics, journalism, etc.

- **Consider a** project involving more advanced data structures from Chapter 20 and beyond (e.g. linked lists, stacks, queues, hash tables/dictionaries, binary trees, etc.)

  <u>For instance, here's a fantasy algorithmics problem.</u>

  In an ancient land, the beautiful princess Eve had way too many suitors. She decided on the following procedure to determine which suitor she would marry. First, all of the suitors would be lined up one after the other and assigned numbers. The first suitor would be number 1, the second number 2 and so on. Starting at the first suitor she would then count three suitors down the line and the third suitor would be eliminated from winning her hand and removed from the line. Eve would continue in this manner. To make things more interesting, Eve decided to randomly select one removal, occasionally. She is also trying to decide on a competition to thin the crowd of suitors. She's also toying with the idea of employing a dragon.

  Given this information, write a program that utilizes pointers and a circular linked-list of nodes corresponding to the suitors who wish to marry the princess. Simulate the elimination process, finally arriving at the suitor who is selected for marriage!

- **A text-based Game** of your choice.

- **NOTE**, it is also possible to enhance and expand your CS-171 final project. In that case, you must make a good case with me & really revamp your previous code with the more advanced concepts you have learned this semester. Talk to me!