### CS 172 Final Project

### TEAM MEMBERS: .

### GRADE:

|  |  |  |
| --- | --- | --- |
| **CATEGORY** | **POINTS** |  |
| Proposal & Requirements Specification |  | 20 |
| Project Code Implementation |  | 45 |
| PowerPoint & Presentation |  | 20 |
| Individual Team Member Summary |  | 15 |
|  |  |  |
| **TOTAL** |  | 100 |

### FINAL PROGRAMMING PROJECT:

* This is intended to be a **team project**.
* Each group will **present their final project** to the class during the finals session for this class.

***DELIVERABLES***

/////////////////////////////////////////////////////////////////////////////////////////////////////////

* **Nov 19:** **Project Proposal. Requirements Specification.** Essentially the design for your project – IMPORTANT!
  + List all team partner(s).
  + A **clear problem definition**. State the requirements – what is required of the system? What must it accomplish or provide? Do you make any assumptions?
  + **Specifications**. The design of your project. **Complete UML diagram** complete with classes, designated member functions, member variables and relationships between classes. What classes, structures, variables, functions, etc. will be required?
  + How will you begin to approach the problem? What are the anticipated challenges?
  + Identification of **individual team member tasks**

///////////////////////////////////////////////////////////////////////////////////////////////////////////

* **Due Finals Day: Computer Science II Section 1 (afternoon): Final Tues, December 10, 1-3 pm**

**Computer Science II Section 2 (morning): Final Thurs, December 12, 8-10 am**

**Project Implementation**

* As you develop your code, use **quality software engineering** practices.
* Design the requirements and **UML class diagram BEFORE beginning to code**.
* You should **employ a number** of the following concepts we have learned as appropriate: templates, vectors, pointers, dynamic memory, formal search algorithms, file management, perhaps even inheritance, polymorphism or recursion. There should be a level of evident code sophistication.
* Split up the implementation into portions and tasks **per team members**.
* As you would be required on a job in industry – **Use a comment line to note the author (developer) and date for blocks of code (as you write the code).**  //S. Mabry 11/15/13
* **Consider** reaching beyond into the latter chapters and exploring other more advanced data structures of the STL Library (e.g. linked lists, stacks, queues, binary trees, etc.)
* Your project implementation **MUST** include the following:
  + Object-oriented programming with **classes** and **inheritance**
  + **Separate headers, implementation files** and a **main driver**
  + **Well-commented code, “pretty code” (indentation and spacing to make it easily legible)**.
  + **Sections commented with the developer’s name**
  + **Well-tested code.**
* **Project Presentation & Demonstration**
  + 5 – 10 Minutes. Presented in a professional, polished style
  + PowerPoint to include:
    - Problem definition
    - Itemized requirements and challenges
    - Class hierarchy diagram (or UML Class Diagram)
    - Brief overview of general solution approach
    - Distinctive aspects of your approach (the “cool” factor)
  + Brief demonstration of working program
* **Individual Team Member Summary** 
  + Due in **hard-copy** form, an individual participation write-up from each student.
  + What did you contribute to the project?
  + How did the team work together?
  + What did you learn from the project?
  + What you would do differently another time?
* **Final Project Folder**In each of your CS1 folders you MUST create a folder called FINAL\_PROJECT. In this folder place each of the following items:
  + **Copy of the proposal**
  + **Copy of the project requirements**
  + **Project code**
  + **Presentation PowerPoint document**

**//////////////////////////////////////////////////////////////////////////////////////////////////////**

### What to do? … Some Project Ideas

The following are some ideas for final projects. **You are not limited to these ideas.** Be creative and bring ideas to your instructor! 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 texts for more ideas.

* **NOTE**, it is possible to enhance and expand your 171 final project. In that case, you must reconsider your design and implementation methods in light of more advanced concepts you’ve learned in 172. However, sometimes that makes for a great project. You can redesign and convert into new methods yet take advantage of basic graphics and of your thorough understanding of the problem. Plus it’s almost always possible to add extra features, to strengthen the application. If that is a consideration, talk to me!
* **Text-based or graphical model/simulation** – Models make for a great project because they present real, practical problems easily visualized yet with complexity. Implement a program that simulates a real-life scenario. Examples are simulating an airport air traffic control scheduling, processes of a hospital, a manufacturing floor, movement of aid for a natural catastrophe,
* **Interactive tutoring project** – Implement a program that creates an interactive tutoring for children or high school students. Include some degree of “intelligent” tutoring – in other words, adjust your questions or difficulty of questions according to user responses. Games with educational purposes.
* A **Business application** - For instance, a system for a small business might use a number of files and calculate profits (as per purchases, sales, expenses), might calculate payroll, keep track of bank accounts, etc.
* **Explore** other advanced problems in your text or other texts or in texts that I can make available.

Consider aproject involving more advanced data structures from Chapter 20 and beyond

(e.g. linked lists, stacks, queues, binary trees, etc.)

* **A text-based or graphical** **Game** of your choice. QUALIFIER: C++ graphics is challenging.

The same set of How-to graphics slides from 171 are available. The optional text on the syllabus provides a number of game possibilities, including good tutorials and extensive code samples for graphics. You can use one of those games as a starting point or a source of ideas, to then make it your own. We’ll also have an introductory tutorial on integrating Open GL GLUT graphics.

* **An Interdisciplinary Application**. For instance, a physics model, a bioinformatics model, a biological model, a societal or anthropological issue, scientific problems, an economic model.

***Grading Criteria for Code Implementation:***

|  |  |
| --- | --- |
| 20% | **Problem analysis**: Does the design for the implementation solution exhibit a general understanding of the problem definition? |
| 10% | **Degree of problem difficulty**. 5% would represent a typical 2 or 3 star programming exercise in our text. |
| 40% | **Execution and Results**: Does the project **work** correctly and generate the correct answer(s) according to the problem specifications? |
| 20% | **SW Development Quality**: Does the project utilize appropriate and effective programming concepts? Is the project well designed, contains comments and contains clean, well-organized code. Is the code readable and easy to follow? Commented name of developer for blocks of code |
| 10% | **Efficiency and Creativity:**  Top projects designs show ingenuity, creativity, and efficiency. Demonstration that you have gone **above and beyond** our basic course coverage (i.e. overloaded operators, recursion, data structures, polymorphism, others. |