Background for Programming Assignments
CS 202 Programming Systems

Vision for all CS202 Assignments:
In CS202, all five of our programs will focus on developing object oriented solutions to applications. Therefore, unlike CS163 were we focused on Data Abstraction, now we will turn our focus to developing complete applications using a set of classes working together to solve the problem. The key idea is to break down the problem into small pieces – maybe building blocks – and assign those responsibilities to individual classes. Then, as a team these set of classes create an Object Oriented Solution. You will notice that the assignments are practical problems that face the real world. In reality, each of these problems could take many "man-months" to create. Your job will be to create an application program using a sequence of classes to show a solid understanding of Object Oriented and data structures. Focusing on just the data structures is not enough. Think this term about the application and how it could be set up if this was a real-world application. You will want to focus on how to design classes that are well structured, efficient, that work together, and where each class has a specific **"job"**. **Structs are not OO. Public data is not OO.** ***Points will be deducted if either of these are used.***

Every program this term will need the following constructs:

1. Every program must have 5 or more classes
2. Class designs must include single inheritance hierarchies
3. Each class that manages dynamic memory must have (a) constructor, (b) destructor (C++), (c) copy constructors (C++). Starting with Assignment #3, each class must have an assignment operator as well Member functions with arguments; pass class types as constant references whenever appropriate.
4. No use of structs. (The more you use structs, the less OO your results will be!). Examine the lab .h files to learn how to use classes for node implementations.
5. No classes can exist that have only "set" and "get" member functions, with the exception of a "node" class
6. Minimize the use of "get" functions in general. If you implement a "get" function, first ask yourself why the function exists and if a helper function

can be implemented instead to perform a specific task. Justify your use of getters in your design writeup.

7. All data members must be private or protected (never public).
8. All arrays must be dynamically allocated with new
9. Global variables/objects are not allowed; global constants are fine
10. **Do not use the String class! (use arrays of characters instead!). You may use the cstring class**
11. Use modular design, separating the .h files from the .cpp files. Remember, .h files should contain the class header and any necessary prototypes. The .cpp files should contain function definitions. You must have at least 1 .h file and 2 .cpp files. **Never "#include" .cpp files!**

Whenever you write a class in CS202 – you need to ask yourself "What is the purpose of this class". If it doesn't have a reason to exist, then it probably shouldn't be a class. Or, if its responsibilities are too broad, then it should be broken down into further classes. This is key. Each class should have a specific job. *The responsibilities should be the public member functions and the data that they work with should be the private (or protected) data members.* The classes that you design should work in conjunction with one another – not in isolation of one another. This represents a big change. This means that you will really need to focus more on the application than you may have done in CS163. In fact, you no longer need to limit I/O to the client program…instead you will want that "job" to be done where it makes the most sense! If you find you are writing a class that is always using "set" and "get" functions of another class, ask yourself why isn't that other class doing the job to manipulate the data for you?! This is OOP.

<u>Guidelines for all CS202 Programs:</u>

To get full credit for the assignments, you will need to:

1. Turn in your OOP design first; the due date is different than the programming assignment. Minimum 600 words. Refer to the Style Sheet!
2. Turn in an analysis of your design with the programming assignment; it should analyze your solution in terms of achieving the OOP objectives. Minimum 400 words.
3. Turn in a written discussion of a unix debugger in terms of how it assisted code development. Make sure to use a unix debugger with each project. Minimum 400 words.
4. Program using a consistent style of indentation, header comments for each function, inline comments for each major block of code.
5. Make sure your name is in each file and that there are header comments describing the purpose of the classes, functions that exist within that file.

6. Submit an electronic copy of your .cpp file as attached files to the dropbox on D2L (go to: http://d2l.pdx.edu/ to login). Make sure to hit the submit button after uploading your files (otherwise they will be lost)
7. As a backup, email your work (as attached file(s)) to karlaf@pdx.edu

When Analyzing your Solution – Answer these questions:

1. How well did the data structure perform for the assigned application?
2. Would a different data structure work better? Which one and why...
3. What was efficient about your design and use of the data structure?
4. What was not efficient?
5. What would you do differently if you had more time?
6. **How well your program meets the goals of being Object Oriented:**
    a. Were there classes that had clear responsibilities?
    b. Did one class do the job that another should have? (e.g., is a list class string comparing the underlying data still?)
    c. Where did hierarchical relationships fit in and would it be effective in a larger application?
    d. What was not Object Oriented?
    e. Can you envision a design that would have been more Object Oriented (if you had more time)?