Background for Programming Assignments
CS 302 Programming Systems

Vision for all CS302 Assignments:
In CS302, all five of our programs will focus on developing solutions to larger scale problems using abstractions. This means every solution will have a set of class objects that are designed to work together. 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 – such as building blocks – and assign those responsibilities to individual classes. Then, as a team these set of classes create a solution. We will experience using abstractions using a variety of programming methodologies this term, including generic, functional, and object oriented programming.

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 the programming methodologies 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"**.

Another difference you will notice from CS163, Data Structures, is that we now need to learn how to integrate our own code with existing libraries. It isn't always appropriate to implement the data structures from scratch. Therefore, it will be important to become fluent in working with the Standard Template Library (STL) in C++ and use abstractions provided with Python. Pay close attention to the assignments for instructions of moving from char *'s and the cstring library to the STL's string class, in C++. By Program #3 you should have experienced the STL with lists, arrays, and vectors and be fluent programming with the String class in C++.

To gain proficiency in programming methodologies, all programming assignments must follow these guidelines:

1. **The only place structs may be used is in C++ with exception handling (for data types to catch).**
2. **Every** program must have **5 or more classes**
3. No public data is allowed in C++ or Java
4. Data members (instance data) in Python must be treated as if they were hidden and not accessed directly from outside of the class
5. Class designs must include **single inheritance** hierarchies
6. **Each** class that manages dynamic memory must have (a) constructor, (b) destructor (C++), (c) copy constructors (C++).
7. **For data structures** that you are required to implement, provide a full implementations of insert, remove, remove_all, display, and retrieve unless otherwise indicated in the assignment. **All repetitive** data structures algorithms must be written using **recursion.**
8. When designing member functions, think about where the member functions will get the data to work on. If they do not have arguments, then the only option is the user which may not be appropriate in all situations.
9. In C++, each class that manages dynamic memory must have an assignment operator, copy constructor and destructor at a minimum.
10. Pass class objects as constant references whenever appropriate.
11. No classes can exist that have only "setter" and "getter" member functions, with the exception of a "node" class.
12. Minimize the use of "getter" 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.
13. All data members must be private or protected (never public).
14. All arrays must be dynamically allocated with new
15. Global variables/objects are not allowed; global constants are fine
16. **Do not use the String class! (use arrays of characters instead!). You may use the cstring functions**
17. Use modular design, separating the .h files from the .cpp files.
    a. Never implement member functions within a .h file.
    b. The .cpp files should contain all function definitions.
    c. Create a **readme** file to explain how the grader should be compiling your work
    d. Keep the grading process in mind. **Do not have over FIVE .cpp files.**

Whenever you write a class in CS302 – 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 how we will get the most from our abstractions!

Guidelines for all CS302 Programs:

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

1. 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.
2. 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 200 words.
3. Program using a consistent style of indentation, header comments for each function, inline comments for each major block of code.
4. 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. I expect a **block of comments** with each file!
5. Submit an electronic copy of completed program as an archive to Canvas

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)?