STYLE SHEET
CS 202: Programming Systems

Three documents will be created for each programming assignment. They are described as follows:

I. Written Design (prior to programming):

- **Prior to each program you must provide a written design (600 word minimum).**

- The design writeup has a separate due date and may **not** be turned in late

- This must be written in English using complete sentences.
  1. 600 word minimum
  2. It should cover the major design considerations
  3. Discuss what classes you are intending to create
  4. Discuss the relationship between those classes (using, containing, hierarchical)
  5. Discuss what methods are needed to avoid excessive use of "getters"
  6. Outline the functions that you need for each class and how they will be used by other classes in your design
  7. UML diagrams are highly encouraged

II. Analyzing your Design (after programming):

- **With each program you must provide a written analysis (400 word minimum).**

- This must be written in English using complete sentences.
  1. 400 word minimum
  2. Discuss the effectiveness of your design and classes
  3. Discuss the validity of your approach
  4. Analyze your results in terms of object oriented programming methodologies
  5. What major changes did you have to make in your design and explain why
  6. Describe the efficiency of the approach and the efficiency of the resulting code
  7. **Think in terms of analyzing your solution!**

- For the **data structures** reflect on 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?

• **Consider how well your program meets the goals of being Object Oriented:**

1. Were there classes that had clear responsibilities?
2. Did one class do the job that another should have?
   (e.g., is a list class string comparing the underlying data still?)
3. Where did hierarchical relationships fit in and would it be effective in a larger application?
4. What was <u>not</u> Object Oriented?
5. Can you envision a design that would have been more Object Oriented (if you had more time)?

III. Debugger Writeup (after programming):

• **With each program you must provide a writeup of how you used a linux debugger such as gdb (400 word minimum).**

• This must be written in English using complete sentences.

1. 400 word minimum
2. Discuss the effectiveness of the debugger
3. Discuss what problems it helped you solve
4. Did you discover how it could be used to enhance the programming experience
5. Discuss features that you would like to learn about so that you could use them the next time you program
6. Discuss the validity of your approach

Style Requirements for Source Code
Requirements for internal documentation in the form of comments are listed below.

1. Provide detailed header comments
   a. A heading explaining what the program does and listing the name of the program author, date, class number and program number.
   b. A heading must be supplied indicating the purpose of the entire program; in addition, each separate function should have a heading describing it purpose and arguments.
   c. Expect to write about a paragraph for each file. A .h file should have header comments discussing why someone would want to use the classes and how to use them. A .cpp file should discuss the data structures used and algorithms.

2. Each file should also have a heading, explaining the purpose of that module <u>and</u> the listing the filename (this is important!).

3. A comment following each variable definition telling what it is used for.

4. Comments to explain any program action whose purpose is not obvious to anyone who reads the code.

5. Use mnemonic names for identifiers that relate to their purpose.

6. A consistent pattern of indentation. See the attached C++ style requirement for examples.

7. White space (blank lines) to separate functions.

8. For each function, explicitly list the input data and the output that will result from that function. Make sure to include a header comment for each function; this should explain the purpose of the function as well as describe the arguments.

This means that there <u>must</u> be a header for each function <u>definition</u>. Even for each <u>member function</u>!

The following should be avoided.

1. NEVER us global variables in these programs!
2. Avoid the use of exit, continue, or break to alter the flow of control of loops
3. Avoid using while(1) type of loop control
4. Avoid using the string class – instead use arrays of characters

Working with Multiple Files

1. All projects will have multiple files; make sure to zip the files together so that only one upload occurs to D2L. This is important because D2L changes the file names unless they are zipped. *Ask the lab assistants to demonstrate the process!*

2. Header files (.h files)
-- comment the beginning of this file with:
a) your name, class number, project number, name of the file
b) a description of what this file contains (purpose of the header file)
c) structures, classes, prototypes definitions

3. Implementation files of your classes (.cpp files)
-- comment the beginning of this file with:
a) your name, class number, project number, name of the file
b) a description of what this file contains (purpose of the header file)

-- place the code in the following order (for a class' implementation):
a) constructors -- default, ones with arguments
b) destructor (if you have one)
c) remaining member functions

-- if you destructor calls another function -- then place that function immediately after the destructor

4. Implementation of the main and any other .cpp files

INSTRUCTIONS TO Submit PROGRAMS: IMPORTANT NOTES:

1. Submit your program to the D2L Dropbox for the appropriate assignment.

2. If you submit your program to the wrong dropbox, it will **not** be graded correctly

3. It is best to wait to submit your program to the D2L dropbox until you are satisfied. Only one version of your program will be graded.

4. If you submit a program to the D2L dropbox after the late due date, the grader is under no obligation to grade the assignment unless you have instructor approval.

5. With D2L, you must first upload the file(s) and then select the "**Submit**" button. Forgetting to do so will cause the uploaded files to be lost

6. As a precaution, **ALSO** email your programs to: karlaf@cs.pdx.edu The grader will be using the D2L dropbox, but if there is a problem with D2L, emailing your programs to karlafgr will ensure that you get full credit, as appropriate.

   Always keep a backup of your programs!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!

10. **REMINDERS:** Every program must have a comment at the beginning with your first and last name, the class (CS202), and the assignment number. This is essential!