Program Documentation and Submission Guidelines

- **This document provides additional information for use with programming assignments this term. See the Grading Guidelines Sheet under Canvas for additional information.**

General Overview

- The topic breakdown of the program sets are as follows:
  - PS 1- COSC 1436/COSC 1437 Review
  - PS 2- STL, Stacks, Queues, Linked Lists
  - PS 3- Recursion, Trees, Sorting, Graphs
  - Make Up- Comprehensive

The topics in PS 1, PS2 and PS 3 will be relevant to the topics covered in the textbook or in notes.

Program Documentation

- All code submitted must follow the coding conventions from previous classes and modifications as specified below. Good style is what makes a program readable.
  Every program in the program set should be begin with a leading block comment header containing:
  - Your Name
  - COSC 2436
  - Program Set Number
  - References

- If code is borrowed from other sources- cite it in the comment block. If you find it necessary to obtain help from someone else in completing your assignment, indicate it in the comment block. There is nothing wrong with using the code from an existing program as a template for a new program. One should, however, understand the code and give credit where it is found.

Program Submission

For each program set:

- Create a folder. Name the folder with your last name. Inside the folder add the following:
  - The applicable source code files (`.cpp`) for each problem. Include any extra credit. Do not include any project files or data files.
  - Sample runs for each problem showing actual input and all values output including all test cases in a text file or a screen shot in a `.docx` (Word) file.
- Zip the named folder from above. "Zipping" files (`.zip` format) is a way to combine multiple files into a single file that can be easily uploaded and downloaded. Do not use `.rar` or any format other than `.zip.`
- Submit the single `.zip` file under the appropriate location in Canvas. Your submission should consist of multiple files in the named zipped folder.

Notes:

- **The instructor will run individual programs to make sure the programs work. Remember your programs will be tested using the stated language compliers. It is your responsibility to use the correct compiler and submit the correct files.**
- **The instructor really does not need to grade your assignment- you know how you did. Do not ask the instructor to check your assignment before turning it in for a grade. There should be no major surprises if points are lost for a problem.**

Other General Guidelines

- Keep the code straightforward, simple, elegant, and follow any data structure requirements.
- If the problem does not specify a data structure/format- then implement the program whichever way you wish. If you have a choice between writing code that runs quickly and writing code that is easy to understand, make it easy to understand.
- If the problem says, "Refer to the sample output below." that means the output in your program should be similar if not exactly like the example given. You have a little freedom but stay as close as possible.
- Input/output should be user friendly- do not leave anyone guessing on what to do.
- Your program test data should not be hard coded in your program (unless specified in the problem to do so). The test data should be inputted from the keyboard or from an input file as specified in the problem.
- The instructor will test your program with other data sets than those examples given with the problem. Those additional data sets will not be provided to you.
- Do not modify any data files if given to you by the instructor to be used for a specific problem.
- Always explicitly open, close, and check for valid files. All data file names must be entered from the keyboard.
- Do not use `break`, `continue`, or `goto` statements, unless the `break` is used within a `switch` statement.

C++ Specific

- No programs will be written using graphics.
- For problems requiring the use of strings, always use the built-in C++ class `string`- do not use C-style strings.
- Place multiple classes into one file. Have one `.cpp` file with one or more classes in a file.

```
#include <iostream>
using namespace std;

class BaseClass
{
public:
   BaseClass()  // Constructor
      { cout << "This is the BaseClass constructor.\n"; }

   ~BaseClass() // Destructor
      { cout << "This is the BaseClass destructor.\n"; }
};

class DerivedClass : public BaseClass
{
public:
   DerivedClass()  // Constructor
      { cout << "This is the DerivedClass constructor.\n"; }

   ~DerivedClass()  // Destructor
      { cout << "This is the DerivedClass destructor.\n"; }
};

int main()
{
   cout << "We will now define a DerivedClass object.\n";

   DerivedClass object;

   cout << "The program is now going to end.\n";

   return 0;
}
```

- In a Visual C++ console program, if the output window closes quickly add the following as the last statement in the `main()` function:

```
cin.get();  //Hold the output window
```

- When creating your project in VC++ select an empty project. Otherwise, the compiler will add additional code that will likely not let your program compile.
- Eliminate all warnings with the VC++ compiler. Most compiler warnings are easy to correct.
- Do not include the following directives:

```
#include "stdafx.h"

OR

#include <bits/stdc++.h>
```

  These are non-standard C++ header files and are non-portable. Using these could cause your programs not to compile when grading.
- Inputting the data file name from the keyboard- Gaddis Chapter 5.