COSC 1437 Grading and Guidelines (DL)-Fall 2016

- **This document provides additional guidelines used this term. See Course Syllabus/ICR under the MyTCC portal for more information.**

Course Grading

Programs:

- There will be 6 programming sets given this term. You may choose one of two options for each program set.
    - Option 1: Complete two of the four problems (2 of 4) in each program set that counts toward the particular program set grade.
    - Option 2: Complete the one double stared (**) problem in each program set that counts toward the particular program set grade.
- Option 1 is the default choice. Option 2 must be stated in the program comments. If not stated, it will be assumed you chose Option 1. Extra credit can be earned with either choice.
- Each individual programming problem will be worth 10 points each and partial credit will be given. Points for each problem will be divided into the following categories:

Category	Value	Score
Correctness	**5**	Code meets all requirements in specification and functions correctly. Includes elimination of all warnings.
Testing	**2**	Test cases cover all the program specifications.
Style and Quality of code (efficiency)	**2**	Easy to understand solution/efficient. Readable. Follows basic rules of structured/object-oriented programming.
Documentation	**1**	Correct documentation and coding standards followed in all cases. Includes proper program submission directions/output.
Total Points	**10**	

- **A problem that does not compile or suffers a runtime error automatically receives 5 points. No additional points will be awarded.**
- **A problem submission without a `.cpp` file will automatically receive 5 points- submitting the project file only.**

- The individual program grades are calculated as follows:

(total points earned / total point value) * 100

Example: 19/20 = .95 * 100 = 95

- Programs will be submitted via the appropriate location on the MyTCC portal site on/before the stated due date.

- For each program, only the work submitted to the appropriate location will be graded. This includes submitting previous programs not originally submitted. If something was failed to be submitted it will not be graded.
- Submitting programs from other programming classes or sections will not be graded as well.
- Two submission attempts per program will be given. The latest submission will be used for grading; feedback will not be provided on earlier submissions.
- If any problems were submitted late, then the whole program set is considered late. Programs submitted late will follow the same grading/submission standards as stated above. For late work only, if the original grade is 50 or less- the late penalty will be waived.
- No e-mails or printouts of your programs will be accepted. No group work allowed.
- **Extra credit:**
    - Option 1:
        - If additional problem(s) are implemented in a particular program assignment - up to 10 points extra credit can be earned on the particular program grade (5 points per problem).
    - Option 2:
        - Only one additional problem in the set may be implemented for up to 10 points extra credit on the particular program grade.
    - The problem(s) must be clearly marked as extra credit in the comment header; otherwise no extra credit will be awarded.
    - Extra credit is graded on the same program scale listed above (scaled in half).
    - Extra credit will be added after the particular grade has been calculated and extra credit does not carry over from any program set.

Quizzes:

- There will be 4 quizzes given this term.
- All quizzes will be worth a total point value. Each question will have individual point values as well.
- The individual quiz grades are calculated as follows:

(total points earned / total point value) * 100

Example: 19/20 = .95 * 100 = 95

- Two submission attempts for each quiz will be given. The latest submission will be used for grading; feedback will not be provided on earlier submissions.
- Not following any of the directions on the quiz will result in a 5-point deduction on the quiz. The points will be taken off after the grade has been calculated.
- No e-mails or printouts of your quizzes will be accepted.
- Solutions will be provided once the grade has been posted.
- **Extra credit:**
    - If additional problem(s) are implemented on a particular quiz- up to 10 points extra credit can be earned on the particular quiz grade.
    - Extra credit is graded on the same program scale listed above.

- Extra credit will be added after the particular grade has been calculated and extra credit does not carry over from any quiz.

Program Documentation Guidelines

- All coding submitted must follow the coding standards found in the textbook and the modifications as specified below. The book (Gaddis) follows standard C++ programming coding guidelines and practices good habits learned in beginning programming.
- Every C++ program turned in should have a leading comment block that identifies the author, course, program number, and any references used. All programs/projects must have leading comment header that looks similar to below:

```
// Author: James Bond
// Course: COSC 1437
// Program 1
// References/Option:
```

- If code is borrowed from other sources- cite it in comments (references). In programming one does not always have to "reinvent the wheel" each time you program. One should however, understand the code and give credit where it is found.

Program Submission

- **For each programming problem:**
  - Submit only the applicable `.cpp` files only (source code).
  - Capture the output from at least two different test cases (actual output) and either submit a text file or a screen shot of the output in a `.doc` file.

Put all parts of each problem (2 parts above) into one program set folder. Name the folder with only your last name. Zip the folder and submit to the appropriate location under the MyTCC site.

- See sample submission folder under MyTCC site.

Other Notes:

- **A student should have an idea if a submitted program works or not. So there should be no major surprises if points are lost for a particular problem.**
- **The professor will actually run individual programs to make sure the programs do work. Make sure the output file/source code submitted is correct!**

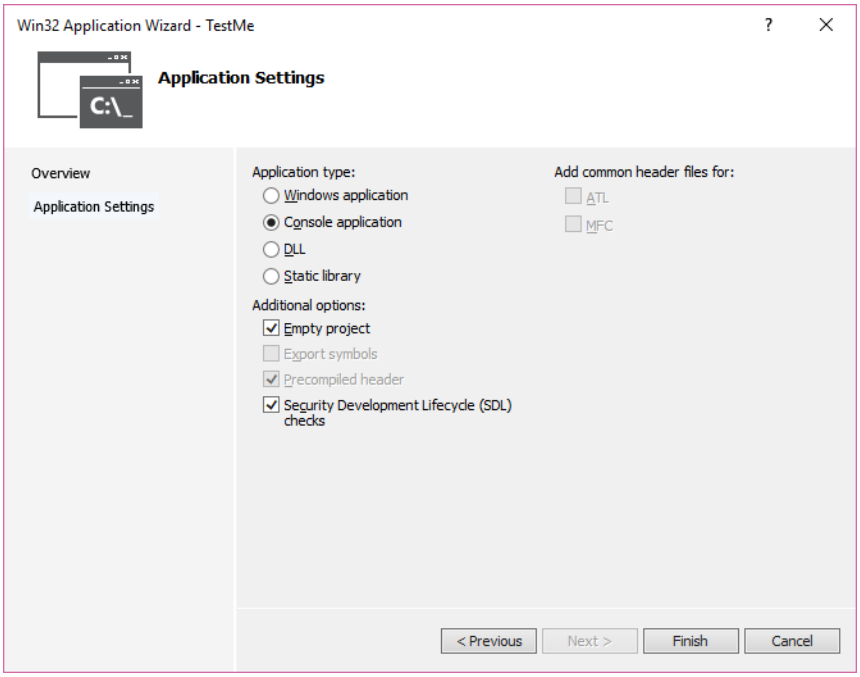
Other General Guidelines

- If the professor does not specify something in the problem -then implement the program whichever way you wish. Suggest using the simplest code/way possible.
- If the problem says "Output should look similar to 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.
- Keep the code straightforward, simple and elegant, and follow any particular data structure requirements.

- Input/output should be user friendly- do not leave anyone guessing on what to do.
- Do not modify a data file in any way if given to you by the instructor. Other relevant data files may be used to test your program for correctness.
- Always explicitly open, close, and check for valid files. Even if the problem directions do not state as such.
- In a Visual C++ console program, if the output window closes quickly add the following statement as the last statement in the `main()` function:

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

- When creating your project in VC++ make sure you select an empty project. Otherwise, the IDE will add additional code that will likely not let your program compile.



- Place multiple classes into one source file. This means there is only one `.cpp` file with one or more classes (`.h` or `.cpp`) in the project.
- Inputting the data file from the keyboard in Gaddis text pp. 283/284.