COMP 2710 Syllabus

Term: Spring 2016
Course: COMP 2710: Software Construction
Schedule: MWF, 9:00 – 9:50 am, Shelby 1103

Instructor: Dr. Xiao Qin (xqin@auburn.edu)
Office: 3101E Shelby Center
Office Phone: 334-844-6327
Office Hours: MWF, 9:50 pm – 10:40 am

TA: Yuanqi Chen, yzc0029@auburn.edu
Office Hour: MWF 2:00pm-2:50pm.
Office: 3139 Shelby Center

Required Text: Savitch, Walter. *Absolute C++*, 5th or 6th Edition. Addison-Wesley.
You may also use: Savitch, Walter. *Absolute C++*, 2nd, 3rd, or 4th Edition. Addison-Wesley.
Reference: Dennis, Wixom, and Roth, *Systems Analysis and Design*. 5th Edition. Wiley.

Course Description

COMP 2710 is highly schizophrenic in that it is both a course on managing the complexity of large systems and an applied programming class. Managing software complexity requires some knowledge of software process. Applied programming means that you will be required to critically analyze real-world types of problems, design algorithms, and then implement those algorithms in high-level code to solve problems. COMP 2710 is as much about learning to solve problems as it is about C++ Programming. This course requires organization, effort, and discipline. You should prepare for every class and bring LOTS of questions – COMP 2710 is not a passive viewing experience. If at any time you feel that you are falling behind, you should contact the instructor immediately and come to office hours frequently. The keys to success in this course are attending every class, starting on homework assignments as soon as they are assigned, actively studying for exams, and always requesting help in a timely fashion.

This course typically requires 9 hours of time per week, on average for the average student. If you don't have it, drop.

Course Objectives
Upon successful completion of the course, the student should be able to:

1. Describe the difference between function-oriented and object-oriented software development
2. Analyze problems to determine system requirements
3. Create data flow diagrams to visually represent the movement and interaction of data
4. Develop object-oriented software designs that map to requirements identified in analysis
5. Grasp both C++ Syntax and Semantics
6. Develop software using sound programming principles
7. Use assertions and design by contract to develop high-reliability programs

8. Understand concepts of data abstraction, efficiency, and memory management
9. Understand how to perform unit and system-level testing
10. Apply a variety of strategies to debug simple programs
11. Use software tools (e.g., gdb, make, argoUML and MS Visio) in support of the software development
12. Describe how to use version control to conduct software release management
13. Demonstrate capability to use runtime configuration to bind variable values and program settings when a program is running
14. Construct, execute and debug programs on Linux
15. Have experience in developing non-trivial software applications

Topic List (not necessarily strictly in chronological order):

1. Administrative Stuff (1 hour)
2. Prepare Software Development Environment (1 hour)
3. C++ Intro: History and Basics (2 hours)
4. Flow of Control (1 hour)
5. Functions (1 hour)
6. I/O: Basic, File (2 hours)
7. Call by References (1 hour)
8. Strings and Streams (1 hour)
9. Maintain Multiple Versions (1 hour)
10. Structures & Classes: Basics (2 hours)
11. Pointers (1 hour)
12. Linked Lists (5 hours)
13. Dynamic Arrays (1 hour)
14. Constructors and Overloading (2 hours)
15. Use Cases (3 hour)
16. Function-Oriented Design (2 hours)
17. Testing (3 hours)
18. Makefile (1 hour)
19. Separate Compilation (1hour)
20. GDB – A Debugging Tool (1 hour)
21. Object-Oriented design and Class Diagram (1 hour)
22. System Sequence Diagrams (2 hours)
23. Vectors and Iterators (1 hour)
24. Namespaces (1 Hour)
25. Inheritance and Polymorphism (1 hour)
26. Review Sessions (3 hours)
27. Exams (3 hours)

Assessment

Exams: Two Midterm Exams, Final Exam
Exams will be closed book, closed notes. Questions will be derived from lectures, material taught only in class, and from assignments. Question format will be mixed.

Short Homework Assignments and Activities: 6 homework assignments
These activities will be take-home in nature and designed to reinforce concepts taught in class. They will be due in writing at the beginning of class. An electronic copy may also be necessary (specified in the assignment). Generally, these assignments are designed to be low-risk in the sense that they are designed to assess thinking and effort, rather than to strictly punish errors.

1. Learn how to use Linux, putty, and g++ (1 week)
2. Write a very simple C++ program (1 week)
3. Learn test drivers, assertions, random numbers (1 week)
4. Use arrays, streams, and file I/O (1 week)
5. Use structures, linked lists, multiple versions, and functions (1 week)
6. Learn how to define and implement classes, use strings and dynamic arrays (1 week)

Individual Construction Projects: 2 project assignments
These projects will consist of the creation of design artifacts (turned in prior to the implementation) and correct C++ implementations of project specifications. All projects should be made to compile under the g++ compiler on Linux. **You may use any development platform or compiler, but your projects will be graded ONLY on a g++ compiler running on Linux. If your project does not work in that environment, you will NOT get credit. Always test it yourself in the lab (Shelby 2119 and Shelby 2122)!**

7. Project 1-Phease 1: Use tools (e.g., the argoUML tool) to create a data flow diagram and a use case diagram (1 week)
8. Project 1-Phease 2: Design a singly linked list, learn the function-oriented design approach, design function prototypes, learn how to design algorithms using pseudo-code, and design unit and system testing cases (1 week)
9. Project 1-Phease 3: Design and implement a non-trivial application using the function-oriented development approach, develop a singly linked list, gain experience with unit and system testing, and develop a reasonably user-friendly application (2 weeks)
10. Project 2-Phease 1: Create a data flow diagram and a use case diagram for a software toolkit (1 week)
11. Project 2-Phease 2: Design learn the object-oriented design approach, design a class diagram, design system sequence diagram, and design testing cases (1 week)
12. Project 2-Phease 3: Develop a software toolkit, build namespaces, use the make utility program, perform separate compilation, and create a single compressed (.tar.gz) file. (2 weeks)

Individual Projects will be graded as follows:
Analysis, Design, and Testing Documents: 30%
Program meets specifications and implements key features correctly: 60%
Adhering to coding style: 10%
(Note that efficiency is not a grading criteria in this class)

I reserve the right to assess other penalties for any errors not strictly covered in the above rubric. I also reserve the right to give bonus points for exceptional work.

Grades:

- Exams 40% (Two midterm exams and one final exam)
- Quizzes 10%

- Homework 20%
- Projects 30%

A [90, 100], **B** [80,90), **C** [70,80), **D** [60,70), **F** [0,60)
Note: In order to pass the class, you must receive at least 60% credit on the Individual Construction Projects and Homework, regardless of performance on exams.

Course Policies

Scaling, Curves, etc: Grades will not be scaled, curved, or adjusted arbitrarily. Some opportunities for bonus points may be provided to the entire class at the instructor's discretion.

Project Due Dates: Projects will be submitted through Canvas. Projects will always be due at 11:55 pm on the due date. Late assignments will receive a grade of zero (0) Deadlines will be made as generous as possible to *a priori* take into account illness, other courses, Acts of God, and nearly all conceivable excuses. If you have a documented illness preventing you from completing your assignment, you may submit all of your partial work and request an extension. **This extension is not automatic**.

Late Submission Penalty

- Ten percent (10%) penalty per day for late submission. For example, an assignment submitted after the deadline but up to 1 day (24 hours) late can achieve a maximum of 90% of points allocated for the assignment. An assignment submitted after the deadline but up to 2 days (48 hours) late can achieve a maximum of 80% of points allocated for the assignment.
- Assignment submitted more than 3 days (72 hours) after the deadline will not be graded.

Rebuttal Period: You will be given a period of a week (7 days) to read and respond to the comments and grades of your homework or project assignment. The TA may use this opportunity to address any concern and question you have. The TA also may ask for additional information from you regarding your homework or project.

Quizzes: The number of popup quizzes in this semester will be anywhere between 8 to 10. If you miss a quiz, **no makeup quiz** will be offered. The two lowest quiz scores will be dropped from the calculation by the end of this semester.

Announcements: E-mail is an official form of university communication. You are responsible for all announcements made in class or electronically. You should read your e-mail at least once a day. If a student asks a particularly relevant question, I may forward the response to the entire class for their benefit.

Special Accommodations: A student in need of special accommodations must bring that need to my attention within the first two weeks of class. The need must be properly documented.

Academic Integrity: Students will be expected to understand and follow Academic Honesty policies in place by the university. All work is to be done **individually**. Students should NOT share any project code or even detailed algorithm information with each other. Your programming code is exclusive to you.

Approved references:
The following constitute acceptable references to help you complete assignments.

- The course textbook is always approved and content may be used without citation.
- My course notes, lectures, and advice I give in my office may be used without citation
- Online general web references are fine, provided you give a citation for the website at the top of your code AND clearly label any lines of code that you use (it should never be ambiguous which lines of code you used from a website)
- Other books/textbooks on the language are fine, but require citations
- You are allowed to discuss broad conceptual ideas (for example, the idea of polymorphism) with other students, but never to share code. If you discuss something with another student (even casually), you should always cite that reference in clear terms.

Unapproved references (these constitute Academic Dishonesty):
This is not a complete listing and cases of ambiguity should always be referred to the instructor for approval prior to use.

- Solution manuals for the text (or the like)
- Websites that sell custom code to individuals
- Code written by others (students or otherwise) for this class or similar classes
- Anything not listed under "Approved References" or approved by the instructor

You MUST document references clearly. If you discuss a project with another student or professor, you should indicate what you discussed and who you discussed it with clearly in the header of your project documentation (and/or code).
For example:
//Xiao Qin
//Project2.cpp
//Dr. Homer Carlisle helped me debug a syntax error in my for loop.
//I used Wikipedia.org in order to learn how a genetic algorithm works.
//I spoke with Bob Smith in the class about identifying objects in C++.

If you don't need any sources for an assignment, clearly state "I did not use any external sources for this assignment" in your source code. Failing to document sources is plagiarism and will be penalized.

If you are unsure whether or not to document a source, it is better to document. Breaches of Academic Honesty will be referred to the Academic Honesty Committee and the strictest sanctions possible (including expulsion and failure) will be my recommendation.

If you are ever unclear about whether or not a course of action is unacceptable, you are always encouraged to consult the instructor.

Attendance: You are responsible for all material and announcements presented in class (even if absent).

1) For exams:
    a. if you have a planned university-approved absence you must make me aware before the test in writing (with appropriate documentation).
    b. if you an unplanned absence, you must provide written, documented, and verifiable justification.
    c. Make-up exams will be different from the original exam.
2) If you are late for a test, you do not receive any extension.
3) Consistent attendance is typically essential to obtaining a good grade (C or better) in COMP

2710.

Reading: Students are expected to read the appropriate sections of the book before each lecture.

Getting Help: Assignments may prove challenging and time-consuming. You are always welcome to bring questions concerning labs to the class, as well as to office hours. A good strategy is to always start early on projects, so that if you run into difficulties, you can get help as soon as possible. I will do my best to answer e-mails concerning labs within 48 hours of receiving them; however, I do not guarantee that I will always have time to debug code via e-mail (I prefer not to do so). For time-consuming problems dealing with code, office hours are always preferable. I will not help debug code via e-mail on the day an assignment is due. The Blackboard Discussion Board is a great way to ask questions so that everyone benefits from the answer to your question!

Office Hours: You are always welcome to drop by during office hours to discuss projects or general concepts. To get urgent help or advice out of office hours, it is recommended to send an email in advance to make an appointment.

Course Difficulty: Typically, the course starts off relatively easy and gets harder as time goes on. Often, students are deceived by the (slower) initial pace and develop lazy habits at the beginning of the course. By mid-semester, they have thrown away many grade opportunities and find themselves in a bad situation with respect to grades. No amount of effort at the end of the class will compensate for consistent, dedicated effort throughout the class. Whether or not you have past experience with programming (or even with C++), my strongest recommendation is that you respect the class and come to class ready to engage every single class period. If you do this, you will dramatically increase your chances of success.