

**CMPS260  
INTRODUCTION TO DATA  
STRUCT & SOFTWARE DESIGN  
SYLLABUS  
FALL 2009**

**Instructor:** Jim Etheredge  
**Office:** ACTR 219  
**Office hours:** See my web site  
**Phone:** 482-6609  
**Email:** [jne1390@louisiana.edu](mailto:jne1390@louisiana.edu)  
**Web Site:** <http://www.ucs.louisiana.edu/~jne1390/>

<b>Class meeting times:</b>	<b>Lecture:</b>	<b>TR</b>	<b>11:00 AM – 12:15 PM</b>	<b>HLG 215</b>
	<b>Lab section 001:</b>	<b>TUESDAY</b>	<b>12:30 PM – 1:45 PM</b>	<b>ACTR 106</b>
	<b>Lab section 002:</b>	<b>TUESDAY</b>	<b>2:00 PM – 3:15 PM</b>	<b>ACTR 106</b>
	<b>Lab section 003:</b>	<b>WEDNESDAY</b>	<b>2:30 PM – 3:45 PM</b>	<b>ACTR 106</b>

**Current Catalog Description:** Integrated software engineering principles, fundamental data structures and algorithm design and development. Focus on requirements, specifications, design and testing. Fundamental data structures will include arrays, linked lists, stacks and queues. Prerequisites: CMPS 150 with a grade of C or better. Co-requisite: Math 270.

**TEXTBOOK:** C++ Plus Programming: Program Design Including Data Structures (Fourth Edition), 2009 by D. S. Malik. ISBN 978-1-4239-0222-5 (It's the CMPS150 textbook).

**COURSE GOALS:**

- ▶ To gain programming experience in C++ on Unix.
- ▶ To gain experience with object-oriented development.
- ▶ To understand software design principles. The documentation standards for this class direct students in the specification and design of object-oriented software systems.
- ▶ To develop a working knowledge of lists, stacks, and queues. The programming projects help students understand the use and implementation of these data structures.

**LEARNING OUTCOMES OF THIS COURSE:**

Upon successful completion of this course, students will have the ability to:

- Understand and apply fundamental programming constructs in C++ (i.e., decisions, branches, structs, arrays, parameter passing, etc.)
- Analyze requirements and design correct, efficient object oriented software solutions
- Implement, test, and debug non-trivial software in C++
- Understand and apply object oriented design principles
- Understand and apply object oriented programming concepts (i.e., classes, methods, operator overloading, templated classes,
- Understand the uses of and the need for common class types (i.e., container classes, iterater classes, etc.)
- Understand and apply pointers in the implementation of object oriented software
- Understand the issues associated with the user of pointers in classes
- Understand and employ linked lists, stacks, and queues in the implementation of object oriented software
- Work in teams

**GRADING POLICY:**

**Scale:**

A	90 - 100%	
B	80 - 89%	
C	70 - 79%	(See NOTE 1)
D	60 - 69%	
F	< 60%	

**Breakdown:**

The programming projects, design project, lab assignments, and the "CMPS150 material test" together will count as: 20% of your grade.

Tests:	Mid-term exam #2	25%	
	Mid-term exam #3	25%	
	Final exam	30%	(FRIDAY, DEC 11 <sup>th</sup> 2:00 pm – 4:30 pm)

**NOTE 1:** YOU MUST HAVE AT LEAST A 65% EXAM AVERAGE AND A 65% AVERAGE OVER ALL PROJECTS AND ASSIGNMENTS IN ORDER TO GET A C OR BETTER IN THE CLASS.

**NOTE 2:** ATTENDANCE IS REQUIRED TO UNDERSTAND THE MATERIAL. MORE THAN 3 ABSENCES (THIS INCLUDES LECTURES AND LABS) WILL BE GROUNDS FOR YOUR REMOVAL FROM THE CLASS.

**NOTE 3:** THERE WILL BE NO MAKE-UPS GIVEN FOR MISSED EXAMS.

**NOTE 4:** AS YOU ARE AWARE, CERTAIN TYPES OF ASSISTANCE ARE INAPPROPRIATE IN THIS CLASS. READ THE DEPARTMENTAL COLLABORATION POLICY IF YOU HAVE QUESTIONS. YOU ARE RESPONSIBLE FOR KNOWING THE DEPARTMENTAL POLICIES. THE *MINIMUM* PENALTY FOR CHEATING IS A ZERO FOR *ALL* PARTIES INVOLVED.

**NOTE 5:** YOU MUST TURN OFF ALL CELL PHONES (AND ANYTHING ELSE THAT MAKES A NOISE) BEFORE CLASS BEGINS.

**ASSIGNMENTS:**

Some programming projects will not be accepted late. For those that are, the late assignment penalty is as follows: one day late – 10%; two days late 30%; three days late – 60%; four days late – 100%. All lab assignments are due at the beginning of the lab. Late lab assignments will NOT be accepted.

- You must use the "CC" compiler to compile your assignments. Do NOT use g++!
- To submit your assignments, they must be in the class account (directory). Make sure that you follow the naming convention established for each assignment. Remember that Unix is case sensitive; that is, upper and lower case letters are different. If you have an emergency and cannot finish the assignment on time, contact me prior to the deadline, not after.
- From the date graded material is returned you have ONE WEEK in which to question your grade. After that time there will be no further discussion or review of the grade.

- You may submit an assignment as many times as you like before the deadline. The latest submission overwrites previous submissions. If you submit your assignment after the deadline, it will be late.
- Do NOT change the access rights to your cs260 account!! If you have a problem during the development of an assignment, we will only be able to help if we have access to your directory.
- Once you submit an assignment, do not modify the file or change the time stamp of the files until you have resolved any questions about the grade you earned for the assignment.
- For most assignments you will submit a manila folder containing documented printouts of your source code (and input data and sample runs if applicable). Do not staple or clip the sheets together. Write your name, CLID, course, section, and assignment number on the folder tab.
- In addition to the standard documentation, each source file for the assignments must include the following certification of authenticity comment at the very beginning of the file.

```
// Certification of Authenticity:
// I certify that all code contained in this program was written by me.
```

## **COURSE TOPICS AND READING ASSIGNMENTS:**

- **Introduction and Review**  
Syllabus, course coverage, grading, Unix Review and other handouts
- **Classes, Object Oriented Programming and Data Abstraction**  
Reading: Chapters 11 & 12 - Read all
- **Pointers, Classes, Virtual Functions, Abstract Classes and Lists**  
Reading: Chapter 13 - Read all
- **Overloading and Templates**  
Reading: Chapter 14 – Read all
- **Exceptions**  
Reading: Chapter 15 - Read all
- **Recursion**  
Reading: Chapter 16 - Read all
- **Linked Lists, Stacks and Queues**  
Linked Lists - Reading: Chapter 17 - Pages 982 – 1018, 1045 - 1064  
Stacks - Reading: Chapter 18 - Pages 1076 – 1112  
Queues - Reading: Chapter 18 - Pages 1131 – 1149
- **Design**  
Reading: Programming Documentation Standard, parts I, II, II.1 - II.3
- **Ordered Lists, Doubly Linked Lists**  
Reading: Chapter 17 - Pages 1018 (Ordered Linked Lists) – 1045
- **Sequential and Binary Search, Asymptotic Notation (Big O Notation)**  
Reading: Chapter 19 - Pages 1184 - 1204