CS 3358: Data Structures
Summer II 2015

Section 751

Instructor: Dr. Jill Seaman
Comal 307G
js236@txstate.edu

Course Webpage: http://www.cs.txstate.edu/~js236/cs3358

Office Hours: MTWR: 12:45PM – 1:45PM
and by appointment.

Meeting Time/Place: MTWRF 2:00PM-3:40PM DERR 234

Open Labs: DERR 231: Linux Lab
MCS 590: Windows Lab

Textbook: Data Structures and Problem Solving Using C++, Weiss, 2nd Ed.
ISBN 0-201-61250-X

List of required readings:
Selected sections from chapters 1-3, 6-9, 16-21

Prerequisites:
C or higher in CS 2308: Foundations of Computer Science II
C or higher in MATH 2358: Discrete Mathematics I

Course Description: A course covering classic data structures and analysis of elementary algorithms, with an introduction to recursion.

Grading:			
	Programming Assignments:	25%	6 total
	Exam 1:	20%	Jul 17 (Fri)
	Exam 2:	20%	Jul 31 (Fri)
	Final Exam (comprehensive):	35%	Aug 6 (Thurs) 5:00pm-7:30pm

Attendance: I record attendance every day and expect you to be in class every day. If you are unable to attend, you are responsible for the material and announcements that were covered that day.

Programming Assignments: The programming assignments involve writing programs in C++. **You will do this work in pairs (with a partner).**

Makeup Policy: Missed programming assignments cannot be made up.
Exams may be made up in exceptional circumstances, with approval from the instructor.

Late policy for programming assignments: see the class webpage.

Notifications from the instructor: Notifications related to this class will be sent to your Texas State e-mail account. Be sure to check it regularly.

TRACS: We will use the TRACS website for the following:

- Grades (Gradebook2 tool)
- Programming assignment submissions and feedback (Assignments tool)
- Resources (code you can use in your programing assignments)

Everything else will be on the class webpage (including lecture presentations and demo code).

Campus Labs: You may use DERR 231 (the Linux Lab) or MCS 590 to work on your programming assignments. You may also use your own computer. You may use any IDE that supports C++ programming.

Withdrawals/drops: You must follow the withdrawal and drop policy set up by the University. You are responsible for ensuring that the drop process is complete. http://www.registrar.txstate.edu/registration/drop-a-class.html

Last day to drop: July 24, 2015.

Classroom Behavior: Do not disrupt other students during class. Be respectful.

Academic Honesty: You are expected to adhere to both the University's Academic Honor Code as described here, as well as the Computer Science Department Honor Code, described here: 2013 0426 HonestyPolicy CSPPS.doc.

- **All assignments are to be done in pairs.** Each pair must **write their own code**.
- Do not include code (or other materials) obtained from the internet in your assignments (except what is provided or allowed by the instructor).
- **Do not email your program to anyone (except your partner or the instructor)!**

The penalty for submitting a program that has been derived from the internet or any other non-approved source will be a 0 for that assignment.

Accommodations for students with disability:
Any student with a special needs requiring special accommodations should inform me during the first week of classes. The student should also contact the Office of Disability Services at the LBJ Student Center.

Course Objectives:

Before of the course, the students should be able to:

i. Develop (implement) C++ programs including multiple classes and arrays of objects.
ii. Read and write C++ code that uses pointer variables and memory operations (new, &, *, delete), including pointers to arrays, structures, and objects and the -> operator.
iii. Create, compile, and run a C++ program (in a unix style command-line environment).
iv. Develop (implement) C++ programs with the source code separated into multiple files, using header (.h) files.

At the end of the course, the students should be able to:

1. State the definition of an Abstract Data Type.
2. Explain the value of ADTs in computer science.
3. Describe the semantics (values and operations) of the following ADTs :
   a. Lists
   b. Stacks and Queues
   c. Trees (binary trees, self-balancing trees, heaps)
   d. Hash Tables
   e. Graphs
4. Perform (demonstrate) each of the ADT operations given an instance of the ADT, including each of the 3 tree traversals on binary search trees.
5. Evaluate which ADT should be used to solve a given problem (and defend the choice).
6. Summarize the advantages and disadvantages of static vs dynamic implementations of ADTs.
7. Implement the ADTs in C++
   a. using both dynamic and static underlying types, (including singly and doubly linked lists).
   b. using header files and implementation files.
   c. using C++ templates (generic programming).
8. Write modular programs in C++ that use the ADTs, including using ADTs from the C++ Standard Template Library
9. Describe elementary algorithms for sorting, searching, and hashing, including linear/ binary search, selection/insertion/bubble/merge and quicksort
10. Perform (demonstrate) each of the elementary algorithms listed above.
11. List (in order of increasing growth rate) the 6 categories of mathematical functions used in analyzing algorithms.
12. Analyze algorithms, including sorting, searching, and implementations of ADT operations, for efficiency
    a. Compute the formula to describe the number of steps a given algorithm requires in terms of the amount of data
    b. Classify the formula according to the 6 classifications of mathematical functions
13. State a definition of recursion
14. List and describe common applications of recursion.
15. Read and write recursive functions in C++.
16. Write programs that are well designed and organized.