Data Structures and Object Oriented Design
CS 104 (4 Units)

Overview
CSCI 104 will teach students the fundamentals of object-oriented programming and design with a focus on the representation and manipulation of data using well-known data structures. This course will use the C++ programming language.

[Note: This syllabus is only a derivative of the actual syllabus on the course website. While most policies below are official, some may be altered slightly. You should refer to the course website for the official policies and schedule.]

Learning Objectives
1. Ability to choose appropriate and efficient data structures and algorithms to solve a problem.
2. Ability to compare data structures and algorithms for efficiency using algorithm analysis and experiments
3. Ability to apply algorithm analysis and knowledge of discrete mathematics to evaluate algorithms and data structures
4. Ability to implement and use linear data structures, including stacks, queues, lists
5. Ability to implement and use search structures and algorithms including binary search, search trees, and hash tables
6. Ability to implement and use other data structures such as (non-search) trees, heaps, graphs, and union-find structures.
7. Knowledge of and ability to implement sorting algorithms and compare their performance analytically and empirically
8. Ability to write readable and maintainable code.
9. Ability to design, document, and implement classes and object hierarchies
10. An ability to solve problems using pointers and dynamically managed memory
11. Ability to write recursive functions and understand when recursion is appropriate to a problem
12. Ability to apply tools and techniques for program correctness, such as unit testing, use of a symbolic debugger, and assert statements
13. An ability to explain computational solutions in person and in writing

Prerequisite
CSCI 103L
[Knowledge of basic Unix usage (ssh, vi/emacs/pico, g++), basic stream usage (cout & cin), control structures (if/else if/else, switch, for/while/do-while, etc.), defining & using functions, basic recursion, pointers, dynamic memory allocation,, basic use of strings, arrays & vectors, structs and classes]

Lecture
T, Th 9:30-11 and 11-12:30
Lab
See Schedule of Classes
Course Websites
Blackboard (blackboard.usc.edu) [Used for grade recording]
http://www-scf.usc.edu/~csci104
https://github.com/usc-csci104-spring2014

Textbook Info
Data Abstraction & Problem Solving with C++, 6th Ed.

Other Materials
You should bring your laptop with you to class each lecture as we will be performing in-class programming exercises on many days. Please install the Linux VM available at:
http://alirami.usc.edu/course-vm.ova

Instructor Info
Professor Mark Redekopp
Office: EEB-222, Phone: (213) 740-6006
E-mail: redekopp@usc.edu

TA & Grader Info
TAs: See website
Course Producers (a.k.a. Sherpas): See website

Grading
The following point structure will be used in determining the grade for the course. Final grade will be based upon the total points received, the highest total in the class, and the average of the class.

<table>
<thead>
<tr>
<th>Component</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>Course/Final Project</td>
<td>15%</td>
</tr>
<tr>
<td>Other Homework</td>
<td>15%</td>
</tr>
<tr>
<td>Midterm 1</td>
<td>30%</td>
</tr>
<tr>
<td>Midterm 2</td>
<td>40%</td>
</tr>
</tbody>
</table>

Reading Assignments
Readings from the book and other sources form the base of the pyramid. These readings contain theoretical concepts, examples and usable code that will be very helpful for all the work in this course.

Homework
• Homework will be assigned roughly once per week. It will be graded, and require substantial work. The average student should expect to spend about 8-10 hours per homework.
• Homeworks will typically contain a mix of programming exercises and "theory" questions about data structures and their implementation.
• Roughly every other homework will contain a piece that contributes toward the class project. As the project progresses, students may find it necessary to revisit and improve their earlier solutions, so good coding practices and documentation are strongly encouraged.
• For policies on collaboration and help on homeworks, please see below.
• Of all homework assignments (but not project assignments), the worst one will be dropped in calculating your grades.
• For each homework, a precise due date will be specified. Each student can submit up to 3 homeworks up to 24 hours late.
Submissions more than 24 hours late will not be accepted, and automatically receive a 0. When you submit homework late, you **must** e-mail your TAs about your late submission, so they know to look for it in your Github account. Once 3 homeworks have been submitted late, subsequent late submissions will not be accepted, and automatically graded as 0. To handle technical difficulties or network issues near the submission time you can submit up to 1 hour late and receive a 50% deduction for the whole homework (or use 1 of your 3 grace days). Please be sure to submit early so you don't need to utilize this feature.

- The programming portion of each homework also needs to be demonstrated and approved by a TA or sherpa during lab. Typically, you will do this by explaining your solution for a few minutes in lab the week following the due date. However, if you had a close-to-finished version during the previous week (and it is clear from the code that it hasn't changed much), then discussing your code in the previous lab's week satisfies this requirement.

- Homeworks will from time to time contain "chocolate problems". Chocolate problems do not affect your grade in the course. They are intended to be significantly more challenging, for students looking for a challenge. Chocolate problems will be marked with a number of chocolate candies. Solutions that are (mostly) correct will lead to the solvers getting that number of chocolate candies. Chocolate problems can be solved in groups of up to 3 students (who will then share the chocolate). Chocolate problem submissions should be separate from the rest of the homework, as they will be graded separately.

**Grading Disputes or Re-grades**

For regrades on homeworks, go to either of the two graduate TAs in office hours. You can also submit a regrade request to the course staff through Piazza, but make sure it is submitted as a private request, and goes into sufficient detail to explain where you come from. For all regrades, there is an expiration date of two weeks after the homework results have been returned. Within those two weeks, you can ask for regrades, or, if you know that you are too busy, register that you will want to come in for a regrade within the next week. We will not accept regrade requests more than two weeks after the return date otherwise.

**Grading Policy**

See Website

**Statement on Academic Integrity**

USC seeks to maintain an optimal learning environment. General principles of academic honesty include the concept of respect for the intellectual property of others, the expectation that individual work will be submitted unless otherwise allowed by an instructor, and the obligations both to protect one’s own academic work from misuse by others as well as to avoid using another’s work as one’s own. All students are expected to understand and abide by these principles.
Scampus, the Student Guidebook, contains the Student Conduct Code in Section 11.00, while the recommended sanctions are located in Appendix A: http://www.usc.edu/dept/publications/SCAMPUS/gov/. Students will be referred to the Office of Student Judicial Affairs and Community Standards for further review, should there be any suspicion of academic dishonesty. The Review process can be found at: http://www.usc.edu/student-affairs/SJACS/.

A few things are clearly fine, while a few are clearly not fine. We are listing some of the most relevant ones here:

- Asking other students for hints or discussing high-level ideas. This is clearly ok.
- Having other students look at your code and help you discover mistakes. This is also ok, though the other student may have to be careful about not copying from you (which would have negative consequences for both of you).
- Asking course staff (instructor, TAs, sherpas) for help, ideas, having them look through code, etc. Clearly no problem; if you are asking for too much help, we'll simply not provide that much.
- Copying code from other students, even if you subsequently edit, improve or change it. Clearly not ok, even if you intend to understand the code before submitting it as your own. This is most definitely plagiarism. We will run software on all submissions in the class to detect instances of copying.
- Looking at other students' code (before having finished your own). This is a gray zone. If you just take away some basic conceptual ideas, then it is fine. If your code ends up resembling the other student's very closely, then it is cheating.
- Looking up concepts, syntax, and basic instructions on how to deal with the topics online. This is clearly ok, as you are learning.
- Looking online for solutions to specific homework questions. Clearly not ok, even if you subsequently edit them. You are trying to use the work of others instead of your own. Clearly cheating.
- Posting in online forums asking people to solve homework questions (or parts thereof) for you. Clearly cheating, for the same reason as the previous one.

Unfortunately, we are aware that (1) there are solutions to many homework questions available on the WWW, and (2) even USC students tend to cheat quite frequently on homeworks. Please help us restore faith in the integrity of Trojans by not being those students.

Policies  Statement for Students with Disabilities
Any student requesting academic accommodations based on a disability is required to register with Disability Services and Programs (DSP) each
semester. A letter of verification for approved accommodations can be obtained from DSP. Please be sure the letter is delivered to me (or to TA) as early in the semester as possible. DSP is located in STU 301 and is open 8:30 a.m.–5:00 p.m., Monday through Friday. The phone number for DSP is (213) 740-0776.

**Emergency Preparedness/Course Continuity in a Crisis**

In case of a declared emergency if travel to campus is not feasible, USC executive leadership will announce an electronic way for instructors to teach students in their residence halls or homes using a combination of Blackboard, teleconferencing, and other technologies.
Data Structures and Object Oriented Design  
CSCI 104 (4 Units)

**Course Outline**

<table>
<thead>
<tr>
<th>Lecture</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lecture 1:</td>
<td>Course Overview – Motivation for DS, Review of C++ Library classes [streams]</td>
</tr>
<tr>
<td>Lecture 2:</td>
<td>Review of Dynamic Memory Management (C++ Interlude 2)</td>
</tr>
<tr>
<td>Lecture 3:</td>
<td>Review of Recursion (Chapters 2, 5)</td>
</tr>
<tr>
<td>Lecture 4:</td>
<td>Linked Lists (C++ Interludes 1, Chapter 4)</td>
</tr>
<tr>
<td>Lecture 5:</td>
<td>Abstract Data Types: Set, List, Dictionary/Map (Chapters 1, 8, 18)</td>
</tr>
<tr>
<td>Lecture 6:</td>
<td>Classes, Templates (C++ Interludes 1), Exceptions (C++ Interlude 3)</td>
</tr>
<tr>
<td>Lecture 7:</td>
<td>Running Time and big-O notation (Chapters 1, 4, 10)</td>
</tr>
<tr>
<td>Lecture 8:</td>
<td>Copy Constructors and Operator Overloading,</td>
</tr>
<tr>
<td>Lecture 9:</td>
<td>Inheritance and Polymorphism (C++ Interludes 1, 2, 4)</td>
</tr>
<tr>
<td>Lecture 10:</td>
<td>Lists revisited [Array lists / vectors] (Chapter 9)</td>
</tr>
<tr>
<td>Lecture 11:</td>
<td>Stacks and Queues (Chapters 6, 7, 13.1, 13.2, 14.1)</td>
</tr>
<tr>
<td>Lecture 12:</td>
<td>Qt &amp; Inheritance</td>
</tr>
<tr>
<td>Lecture 13:</td>
<td>STL (C++ Interlude 7)</td>
</tr>
<tr>
<td>Lecture 14:</td>
<td>Iterators (C++ Interludes 6)</td>
</tr>
<tr>
<td>Lecture 15:</td>
<td>Midterm, no lecture</td>
</tr>
<tr>
<td>Lecture 16:</td>
<td>Search Algorithms, Sorted Lists (Chapters 10, 12)</td>
</tr>
<tr>
<td>Lecture 17:</td>
<td>Sorting Algorithms (Chapter 11)</td>
</tr>
<tr>
<td>Lecture 18:</td>
<td>More Sorting Algorithms (Chapter 11)</td>
</tr>
<tr>
<td>Lecture 19:</td>
<td>Graphs and their uses: a brief introduction (Chapters 20.1, 20.2)</td>
</tr>
<tr>
<td>Lecture 20:</td>
<td>Trees and their Implementations (Chapter 15) and Tree Traversals and Search (Chapter 16)</td>
</tr>
<tr>
<td>Lectures 21--23:</td>
<td>Priority Queues (Chapters 13.3, 17)</td>
</tr>
<tr>
<td>Lectures 24--25:</td>
<td>Balanced Search Trees (Chapter 19)</td>
</tr>
<tr>
<td>Lectures 26--27:</td>
<td>Hash Tables and their Analysis (Chapter 18.4, additional handouts)</td>
</tr>
<tr>
<td>Lectures 26--27:</td>
<td>BFS, DFS, Dijkstra's Algorithm (Chapters 20.3.1, 20.3.2, 20.4.4, additional handouts) plus other Graph Algorithms (Chapters 20.4.1--20.4.3)</td>
</tr>
<tr>
<td>Lecture 28:</td>
<td>Union-Find Data Structures (additional handouts, if time)</td>
</tr>
<tr>
<td>Lecture 29:</td>
<td>Slack/Review for Midterm</td>
</tr>
<tr>
<td>Lecture 30:</td>
<td>Midterm 2</td>
</tr>
</tbody>
</table>