Honors Introduction to Programming and Data Structures

Prof. Darrell D. E. Long
Computer Science Department
Jack Baskin School of Engineering

Autumn 2003

This course presents an honors-level introduction to programming and data structures including the fundamentals of Java programming and basic data structures such as linked lists, queues, trees, hashing, and computational complexity. It combines the material covered in CMS 12A and CMS 12B into a single quarter providing a fast-paced introduction to this material for outstanding students who have some prior programming experience but are not necessarily ready to take CMS 12B. Later programming assignments done in the C programming language.

Contacts

- Instructor: Prof. Darrell D. E. Long, 247 Baskin Engineering, (831) 459-2616, darrell@cse.ucsc.edu.
- Teaching Assistant: none, alas.

Required Reading


Recommended Reading


Course Description

An honors programming and data structures course for outstanding freshman computer science and engineering majors with previous programming experience. This course covers the material usually taught in CS 12A, *Introduction to Programming*, and CS 12B, *Data Structures*. The first three weeks of this class cover material taught in CS 12A: programming and documentation skills, algorithmic problem solving, and programming methodologies. Topics for this portion of the class include, but are not limited to, procedures and functions, conditionals and loop control structures, static and dynamic memory manipulations, and text processing. The remaining seven weeks of this class cover material taught in CS 12B: common data structures and the algorithms associated with each data structure. Topics include big “O” notation; pointers, recursion, and dynamic allocation; linked lists and list processing; stacks, queues, binary trees and binary search trees; simple sorting techniques and simple search techniques.
Grading
There will be weekly homework assignments worth a total of 10% of your grade. There will be weekly programming assignments worth a total of 40% of your grade. There will be a comprehensive final exam worth 50% of your grade. Note: An average of less than 50% on the assignments or the final exam will result in a failing grade regardless of the overall average.

Homework
Homeworks will be questions taken from the books and are intended to make sure that everyone is doing the reading and to test your understanding of the material from each chapter. Homeworks will be graded on a simple three point scale: 0 = unsatisfactory (less than 50% attempted or correct), 1 = satisfactory (full credit), and 2 = outstanding (extra credit).

Programming Assignments
The programming assignments are an important component of the course. One thing to remember: your assignments must work in the environment on the CATS system. You can create your programs anywhere you want, but they must work on our system to get full credit.

Homeworks and programming assignments are due on the date and time specified. Late homeworks and programming assignments will not be accepted except when due to true emergencies. Graded homeworks and programming assignments will be returned as soon as possible, usually within one week.

Attendance
Class attendance is mandatory. I will post homeworks, assignments, and important dates on the class web page, but this is provided as a courtesy and is not always complete. It is your fault if you miss something important because you skipped class. Since we do not have a teaching assistant, Laboratory attendance is not required, but it is a good idea to use this time to complete your programming assignments.

Course Requirements
A summary of the skills required to pass the course:
1. Familiarity with Java, ANSI C, and UNIX tools
2. Basic software design principles: design, documentation, comments, coding style
3. Familiarity with Java keywords, types, and syntax
4. Command of Java programming fundamentals:
   (a) declarations and assignments
   (b) procedures and functions
   (c) conditionals and loop control structures
   (d) static and dynamic memory manipulations
   (e) text processing
5. Ability to use Java to solve basic programming problems
6. Simple linked list manipulation, testing, and debugging
7. Knowledge of linear structures, stacks and queues
8. Recursion and intuitive understanding of asymptotic complexity
9. Knowledge of trees and associated algorithms
10. Knowledge of simple forms of searching and sorting
Core Topics

1. Basic tools: javac, java, cc, make, ...
2. I/O
3. Data types
4. Declarations and assignments
5. Procedures and functions
6. Conditionals and loop control structures
7. Recursion
8. Static and dynamic memory manipulations
9. Basic text processing
10. Classes and Object-Oriented Programming fundamentals
11. Basic pointer skills
   (a) Allocation of objects (structures) and the use of references (pointers) to objects
   (b) Construction and access to elements of linked lists, insertion into, deletion from, and traversal of linked lists
12. Interactive debugging and program tracing
   (a) Assert statements, preconditions and postconditions
   (b) Using print statements to trace programs
   (c) Use of specific interactive debugging tools, breakpoints, execution tracing, dumping values of variables
13. Simple linear data structures
   (a) Stacks and queues
   (b) Linked list implementation of each
   (c) Array implementation of each and resizing of arrays
14. Asymptotic complexity analysis, big-“O” notation
   (a) Informal discussion of the basic concepts
   (b) Intuitive analysis of specific algorithms given in class without any significant mathematical rigor
15. Trees
   (a) Binary trees
   (b) Depth first traversals: preorder, inorder, postorder
   (c) Construction, insertion, deletion of nodes in binary trees
   (d) Implementation of binary trees using references (pointers) and nodes
16. Searching and information retrieval
   (a) Linear searching an array and a linked list
   (b) Binary searching an array
   (c) Construction and searching of binary search trees,
   (d) Insertion and deletion in binary search trees
   (e) Intuitive reference to balanced trees, but no details on the balancing algorithms
17. Sorting, details of one of the $O(n^2)$ sorts
   (a) Insertion sort and possibly bubble sort or selection sort
Optional Topics

1. Expression trees and relation to reverse polish notation
2. Hash tables with collision resolution by chaining
3. Informal discussion of balanced binary search trees
4. Breadth first traversal of trees using a queue
5. Sorting, discussion of better sorts, \( O(n \log n) \)

Academic Honesty

While this really should not be an issue, life in the contemporary university discussing academic dishonesty necessary. The bottom line is that you are expected to conduct yourself as a person of integrity. You are expected to adhere to the highest standards of academic integrity. This means that plagiarism¹ in any form is unacceptable.

As a (soon to be) computing professional, I encourage you to consult the code of ethics appropriate to your discipline. The Association for Computing Machinery, the IEEE and the IEEE Computer Society each have professional codes of conduct. You should take the time to read each of them.

Plagiarism will be assumed, until disproved, on work that is essentially the same as that of other students. This includes identically incorrect, off-the-wall, and highly unusual duplicate answers where the probability of a sheer coincidence is extremely low. Automatic program checkers may be employed to compare programs. All parties to this unacceptable collaboration will receive the same (zero) score. Should you be found to be cheating, at a minimum, you will fail that assignment and a letter will be sent to your Department, the School of Engineering, and to your Provost and academic preceptor. I reserve the right to stronger action should I feel that the situation warrants it.

There are no programming groups in this course. Your work must be your own (web archaeology is prohibited as well). This refers to examinations, written assignments and programming projects.

You may discuss programming projects with your friends, but you are expected to abide by the Gilligan’s Island rule. The Gilligan’s Island rule states that following a discussion of the project, a break must be taken for at least a half hour before coding. Watching a mind-numbing program like Gilligan’s Island on television satisfies this rule. Better yet, read a book unrelated to computer science before beginning to code your program: I recently finished Tuxedo Park, or see my personal web page for other suggestions.

The only thing you may bring to such a discussion is you, and no notes may be taken away from the meeting. The copying of files is strictly forbidden.

A zero score on any of the examinations or on one of the programming assignments is considered grounds for failing the course.

¹plagiarize vt. [< L. plagiarius, kidnapper] to steal and pass off as one’s own (the ideas or words of another) to present as one’s own an idea or product derived from an existing source – plagiarizer n. (source: Webster’s New World Dictionary)