

CS 315H Algorithms and Data Structures

Lectures:	MWF 11:00-noon WAG 201
Discussion Section:	T 2:00-3:00 (#55580) RAs 310
Discussion Section:	T 1:00-2:00 (#55585) RAS 313A
Unique numbers:	55580 and 55585
Class Web Page:	<a href="http://www.cs.utexas.edu/users/lin/cs315h/">http://www.cs.utexas.edu/users/lin/cs315h/</a>
Final exam:	Tuesday December 16, 9:00am

	Professor	Teaching Assistant	Admin. Assistant
Name:	Calvin Lin	Andrew Dreher	Gem Naivar
Email:	lin@cs.utexas.edu	afdreher@cs.utexas.edu	gem@cs.utexas.edu
Office:	ACES 3.424	TBD	ACES 3.422
Office Hours:	TTh 3:00-4:00	TBD	

**Course Objectives.** This course introduces fundamental concepts in programming, object oriented programming, and data structures. This course has a heavy programming component, to be completed using Java, which will attempt to spur creativity and develop a healthy attitude towards software.

**Text.** M. Weiss. *Data Structures and Problem Solving using Java*. 3<sup>rd</sup> Edition. Addison-Wesley, 2006.

**Supplementary Texts.** A good Java reference, such as the following, will also be useful. Arnold, Gosling, and Holmes, *The Java Programming Language*. 3<sup>rd</sup> Edition. Addison-Wesley, 2000.

A book such as the following will help you learn Java on your own: Deitel and Deitel, *Advanced Java 2 Platform: How to Program*. Prentice-Hall.

**Prerequisites.** This course assumes a semester of previous programming experience.

**Schedule.** We'll cover the following topics, roughly in the following order. There will also be assigned reading from other portions of the book that are not listed here.

1. **Introduction.** Course overview, administrative matters.
2. **Java.** Basic goals and concepts. (Chapters 1-2)
3. **Object oriented programming.** Encapsulation. Inheritance. Polymorphism. (Chapters 3-4)
4. **Data abstraction.** Iterators, the Collections framework. (Chapter 6)
5. **Algorithm analysis.** (Chapter 5)

6. **Recursion.** (Chapter 7)
7. **Sorting.** Mergesort, bubblesort, quicksort. (Chapter 8)
8. **Trees.** Traversal. Binary search trees. (Chapter 18-19)
9. **Hash Tables.** (Chapter 20)
10. **Priority Queues.** (Chapter 21)
11. **Balanced Trees.** Splay trees. AVL trees. Red-Black trees. Union-Find trees. (Chapter 22)
12. **Graphs.** Graph algorithms. (Chapter 14)

**Attendance.** Attendance is strongly encouraged. Lectures will often contain material not found in the textbook, and any lecture notes that are provided are not intended as a replacement for the lectures.

Please turn off cell phones and laptops in the classroom.

**Web page.** All announcements and handouts will be placed on either the class web page or the TA's class web page:

<http://www.cs.utexas.edu/users/lin/cs315h>

<http://www.cs.utexas.edu/users/afdreher/cs315h>

**Class newsgroup.** The course has a Unix newsgroup named `utexas.class.cs315h`, which is a good place to communicate with classmates regarding homework and programming assignments.

**Computer Accounts.** You should obtain an account on the PC's in the Painter Microlab (PAI 5.38). You will need these accounts to submit your programming assignments. And, of course, you may use these machines to do your assignments if you do not have your own PC. To obtain an account, follow the instructions on the following web page, and realize that the accounts take a day to become active:

<https://udb.cs.utexas.edu/amut/acut/>

If you do not have access to the Internet, there are machines in the Microlab that will provide access to the above web site.

**Microlab Hours** For your convenience, the labs are open 24 hours a day.

## Grading

Prog. assignments and pop quizzes	45%	approximately 8 assignments
Midterm	20%	Thursday October 16 7:00-8:30pm
Final	35%	Tuesday, December 16 9:00am

Unless otherwise specified, assignments are due by 4:00pm of the due date. Late assignments will be penalized 10% per day. System problems, printer failures, and the like are routine occurrences and are not considered cause for extending homework deadlines. To avoid such problems, start early on your assignments.

Requests for corrections to quiz, programming assignment or exam grades must be submitted to the TA within 3 days of the assignment of the grade. Delay in picking up assignments does not extend this deadline. Note that when a regrade is requested, the entire assignment may be regraded, so your overall score may go down as well as up.

**Programming assignments:** Programs will be graded on clarity as well as correctness.

**Exams:** You may bring *one* page of notes (one sheet of 8.5"×11" paper, written on both sides) to both the midterm and the final.

**Illnesses and Absences.** If serious illness prevents you from handing in an assignment or taking a test, you should contact me or the TA immediately to have any chance of receiving special consideration.

**Office Hours/Open Door Policy.** My door is usually open when I am in, so please feel free to come to my office whenever my door is open. In the rare case that my door is closed, I do not want to be disturbed, so **please do not knock** unless you have an appointment with me. Since my schedule is often quite full, you should feel free to make appointments with me via email if you cannot make it to my office hours.

Email is the best way to reach me. Please do not call me on the phone unless you have a true emergency.

**Administrative Assistant.** My administrative assistant is Gem Naivar, whose office is across the hall from mine. *See Gem when you need to retrieve unclaimed assignments or handouts.*

**How to Succeed in this Course.** Read this syllabus. Read the book *before* coming to class.

This course moves fast, so you are encouraged to keep up with the reading and programming assignments. If you get behind, it can be difficult to catch up. If you have any problems or questions, please come talk to me or the TA *as soon as possible* so that we can help.

There may be times when your background has holes relative to your classmates. In these situations, you should certainly seek extra help, but you will at times be expected to do a certain amount of reading and learning on your own. If you are unwilling to do this, you might consider dropping the course.

There may be some of you for whom parts of this course are old material; if this is true, we encourage you to try some of the more advanced, optional parts of the assignments.

Help each other learn. One of the best ways to ensure that you understand a concept is to explain that concept to another person. (Note that while you are not allowed to help each other write programs, we encourage you to discuss concepts that are presented in class and in the book. For example, a particular programming assignment might ask you to use a particular data structure; you are encouraged to discuss the properties and details of this data structure in the abstract, without looking at each other's code.)

Finally, to succeed in this course, it's very important that you learn to think for yourself. For example, you will find that certain aspects of your programming assignments will be underspecified—it's up to you to think about what the right thing to do is.

**How to Fail in this Course.** Ignore the above advice. Assume you can learn everything from the textbook. Start homework assignments late, and don't even start reading the programming assignments until 3 days before they're due.

**No Whining.** Feedback and concerns about the course are always welcome; legitimate grading errors that are identified in a timely fashion will certainly be corrected; but whining is counter-productive and will only irritate those who hand out the grades.

**Collaboration vs. Cheating.** Understand the difference between cheating and collaboration. Collaboration is encouraged. Cheating will lead to failure of the course.

**Examples of cheating** are many, but include accessing another student's account, looking at someone else's code, copying or downloading someone else's code, or allowing others to copy or access your code. Of course, this means that you should not look on the Internet for code to solve your problems.

**Examples of allowable collaboration** include discussions and debates of *general* concepts and solution strategies. A good way to ensure that you are collaborating fairly is to follow the Gilligan's Island Rule:

**The Gilligan Island Rule**

You are free to discuss a problem with others<sup>1</sup>, but you may not bring from these discussions any written or electronic notes. After the meeting, engage in a half-hour of mind-numbing activity, such as watching a rerun of Gilligan's Island, before you resume work. This rule ensures that you are able to reconstruct what you learned during your discussion using only your own brain.

Always cite your collaborators with a brief explanation of the degree of collaboration (eg. "Susan and I discussed various approaches to testing our code." eg. "I am using the quicksort algorithm described in Chapter 8 of Weiss").

**Code Reuse.** The code you submit should be your own. The only exceptions: You may use code (with attribution) from our primary textbook and from the Java standard libraries.

**Using Outside Learning Material.** Materials from the internet should only be used for educational purposes. Thus, you can read about linked lists (and these examples could well contain code), but you must not copy any code or be looking at any of this code when writing anything that you turn in.

**If you have any doubts about what is allowed, ask the professor or TA.**

---

<sup>1</sup>Of course, the rule about not looking at anyone else's code still applies.