CS 173 Discrete Structures

This course is an introduction to the theoretical side of computer science. It teaches students how to construct proofs, as well as read and write literate formal mathematics. It also does a tour of standard concepts used in theoretical computer science, such as modular arithmetic, sets and functions, relations, trees and graphs, counting and countability, and asymptotic analysis of algorithms.

Registering for the class

This course is typically offered in fall, spring, and sometimes summer terms.

We try to ensure that everyone who needs to take the course can get into it, with CS majors having priority. If you can't register because registration is locked or partly locked or the class is full, consult the current term's course page for information.

If you have an A- or above in the prerequisite courses (or a strong academic record and transfer/proficiency credit for these classes), you may wish to register for the honors add-on to this class: CS 196, section 73. This 1-credit course covers supplementary topics, which vary from term to term. Contact the instructor for details. This package of 173 plus 196 will count as an honors class for the James Scholar requirements. However, if you are in a college other than Engineering, you also need to set up an "Honors Credit Learning Agreement." See the CS academic office for details.

Prerequisites

This course is designed for students who have taken introductory programming (CS 125 or ECE 220) and one term of calculus (Math 220, 221) and received a C- or better grade. Similar courses at other institutions, should be acceptable substitutes. ECE 120 (ECE 198JL) is not sufficient background. Specifically, the course assumes strong fluency with precalculus, and programming experience (in any language) that includes writing recursive functions and manipulating arrays and linked lists.

If you have done well in another class that teaches proof construction (e.g. Math 347/348), or have taken a more advanced math class (e.g. abstract algebra or real analysis) that assumes the ability to write proofs, you should probably take the proficiency exam rather than this course.

Proficiency exam

The proficiency exam is given at the start of each term, typically during the first week of classes or during the following weekend. Schedule and sign-up information is at Computer Science Proficiency Exams. These exams are similar to course final exams. To get a sense of whether you can pass the exam and to prepare for the exam, look through the lecture notes and exam preparation materials on the web pages for recent offerings of the course (see below).

To get proficiency credit, you must score at least a B- on the proficiency exam. This requires being familiar with a reasonable percentage of the specific topics from the course and, also, being able to write acceptable proof and display an understanding of recursion/induction. If you don't pass the exam, nothing appears on your record. You may either take the course (recommended) or re-take the proficiency exam.

You cannot get proficiency credit if you have previously taken CS 173 or CS 374. Check with the CS academic office if you have taken 400-level CS courses with significant theory content, as they may also block credit. In cases where you cannot get credit, consult a departmental advisor about whether you must take a substitute class and/or must to take the proficiency exam to be exempted from the requirement.

Pages for specific terms

Web pages for new terms typically appear right before the start of the term (e.g. in August for fall classes).

Fall 2016  Spring 2017
Fall 2015 Spring 2016
Fall 2014  Spring 2015
Summer 2014
Spring 2014  Fall 2013
Summer 2013
Spring 2013  Fall 2012
Summer 2012
Spring 2012  Fall 2011
<table>
<thead>
<tr>
<th>Spring 2011</th>
<th>Fall 2010</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spring 2010</td>
<td>Fall 2009</td>
</tr>
<tr>
<td>Spring 2009</td>
<td>Fall 2008</td>
</tr>
<tr>
<td>Spring 2008</td>
<td>Fall 2007</td>
</tr>
<tr>
<td>Spring 2007</td>
<td>Fall 2006</td>
</tr>
</tbody>
</table>

*Navigate space*