

Introductory Handouts . . . . .	1
Catalog Description . . . . .	1
Front Door . . . . .	1
Basics . . . . .	1
Books and Other Readings . . . . .	2
Work . . . . .	2
The Course Web . . . . .	3
CSC105 At A Glance . . . . .	4
Class 01: Introduction to the Course . . . . .	6
Defining Computer Science . . . . .	7
Algorithms . . . . .	7
Intellectual Foundations . . . . .	7
Social Issues . . . . .	8
Administrative Issues . . . . .	8
How to Use the Course Web . . . . .	9
Administrative Information . . . . .	10
On Teaching, Learning, and Grading . . . . .	10
Introduction . . . . .	10
My Role . . . . .	10
Grading . . . . .	12
Your Role . . . . .	12
Lecturing . . . . .	13
Favoritism . . . . .	13
Summary . . . . .	13
Academic Honesty . . . . .	13
Citing Program Code . . . . .	14
Disabilities . . . . .	14



# Introductory Handouts

These materials are also available online. The online versions may not precisely match the printed ones, since I update my Web pages regularly. The online versions are the definitive ones.

## Catalog Description

### **CSC105: An algorithmic and social overview of computer science (4 credits)**

A study of core topics and great ideas in the field of computer science, focusing on underlying algorithmic principles and social implications. Topics may include multimedia and hypermedia, networks, architecture, programming languages, software design, artificial intelligence, databases, cryptography, and the theory of computing. Includes formal laboratory work. Prerequisite: None.

## Front Door

Welcome to the Spring 2000 session of Grinnell College's CSC 105, *An Algorithmic and Social Overview of Computer Science*, which is described relatively well in the official blurb. My own take on this course is that we'll be visiting a number of topics in computing from multiple perspectives, primarily the CS techniques used to support these topics and the implications of these topics. While this is not a course in computer programming, we will do a little bit of programming, since it is often easiest to understand an algorithm through its implementation. We will be using JavaScript as our development language. This is the first time CS105 has been taught at Grinnell, so some things are not worked out as well as I might hope, and many things are likely to change.

In an attempt to provide up-to-date information, and to spare a few trees, I am making this as much of a "paperless" course as I can. You may also want to read the basic instructions for using this course web.

## Basics

**Meets:** MTuWF 10:00-11:50 a.m.

**Instructor:** Samuel A. Rebelsky, Science 2427. Office hours: Tu 2:15-4:15, W 3:15-4:15 (also feel free to stop by when my door is open)

**Teaching Assistant** Sarah Luebke. Hours to be determined.

**Grading:** Thought questions: 20%; Labs, attendance: 10%; Risks reactions: 10%; Short stories: 30%; Exams: 30%;

This course has a midterm and a final. Both will be in-class exams.

**Extra Credit:** If you identify other sources to help with the thought questions, I am likely to give you some amount of extra credit.

Throughout the term, I may suggest other forms of extra credit.

## Books and Other Readings

Dewdney, A. K. (1993). *The (New) Turing Omnibus: 66 Excursions in Computer Science*. New York: Computer Science Press.

Our text for the algorithmic side of things. The “excursions” in this text are short articles about different topics in computer science. Some are fairly mathematical.

Forestor, Tom, and Morrison, Perry (1994). *Computer Ethics: Cautionary Tales and Ethical Dilemmas in Computer Science*. Cambridge, MA: The MIT Press.

Our text for the social side of things. While this does not cover all the topics we will cover, it gives a very nice overview.

Rebelsky, Samuel (2000). The CSC105 2000S Course Web.

The hypertext that you are currently reading. All of these materials are optional, but you may find them useful.

## Work

**Thought Questions:** At the end of every class, I’ll give you a question or problem to think about for the next class. At times, I will ask you to reflect on implications that are not discussed in our book. At other times, I will ask you to work out computational problems. You should email me a response by 9 a.m. Responses should generally be about one paragraph long.

**comp.risks:** You are expected to keep up with messages in the `comp.risks` mailing list. Since it appears that our news server is not quite up to date, you should read the digest online or subscribe as specified in the instructions. (Not everything in the digest will be interesting or comprehensible.) Send me a one-paragraph comment on some item in the digest within three days of receiving the digest. When appropriate, we’ll talk about risks in class.

**Short Stories:** As part of your work in the class this semester, you will write three short narratives about a risk in computing. Each narrative should be about one page long, and include four discussion questions. I will have samples available in a few days. You must turn in at least one story before break.

**Labs:** We will do a number of labs in the class to help ground you in the technology of computing. Sarah and I will help you with the labs during class time. If we don’t finish in class, I’d like you to try to finish up on your own. You need not turn in your labs; I will simply check to see that you’ve made a serious attempt at the lab.

## **The Course Web**

The course web for this class is the definitive reference to what I expect to happen each day. Most of my students find it useful to check the course news and the daily handout at the start of each class.

Note that the outlines on the Web are a *sketch* of my plans for the day. I also expect to deviate from those plans when questions or ideas in class lead us in different directions. I'll do my best to make these available the day before class, but past experience suggests that I will have difficulty doing so, and that the outlines will change before class in any case.

I anticipate posting much of your work to the course site. If you have concerns about having your work posted, please let me know as soon as possible.

# CSC105 At A Glance

<b>Week 01: Background</b>			
Monday, January 24, 2000 <b>Introduction to the Course</b> Reading: <i>Introductory Handout</i>	Tuesday, January 25, 2000 <b>Lab: Getting Started in the MathLAN</b> Reading: <i>Handout: Getting Started in the MathLAN</i>	Wednesday, January 26, 2000 <b>Markup Languages</b> Reading: <i>Handout: Abbreviated Guide to HTML</i>	Friday, January 28, 2000 <b>Lab: HTML</b> Reading: <i>Handout: Getting Started with HTML</i>
<b>Week 02: Algorithms</b>			
Monday, January 31, 2000 <b>The Building Blocks of Computer Programs</b> Reading: <i>Dewdney 1 (Algorithms), 17 (The Random Access Machine), 48 (The SCRAM)</i>	Tuesday, February 1, 2000 <b>The Parts of an Algorithm</b> Reading: <i>None</i>	Wednesday, February 2, 2000 <b>Computing Square Roots</b> Reading: <i>Dewdney 21 (The Newton-Raphson Method)</i>	Friday, February 4, 2000 <b>Natural Language Programming</b> Reading: <i>None</i>
<b>Week 03: Analyzing Algorithms</b>			
Monday, February 7, 2000 <b>Analyzing Algorithms</b> Reading: <i>Dewdney 15 (Time and Space Complexity)</i>	Tuesday, February 8, 2000 <b>Sorting</b> Reading: <i>Dewdney 35 (Sequential Sorting)</i>	Wednesday, February 9, 2000 <b>Sorting, Revisited</b> Reading: <i>Dewdney 40 (Heaps and Merges)</i>	Friday, February 11, 2000 <b>Multiplication</b> Reading: <i>Dewdney 25 (Fast Multiplication)</i>
<b>Week 04: Introduction to Ethical Issues</b>			
Monday, February 14, 2000 <b>Background for Ethical Issues</b> Reading: <i>Forester&amp;Morrison 1 (Introduction)</i>	Tuesday, February 15, 2000 <b>Security and Computer Crime</b> Reading: <i>Forester&amp;Morrison 2 (Computer Crime)</i>	Wednesday, February 16, 2000 <b>Encryption</b> Reading: <i>None</i>	Friday, February 18, 2000 <b>Public-Key Cryptography</b> Reading: <i>Dewdney 37 (Public-Key Cryptography)</i>
<b>Week 05: Implementing Algorithms in JavaScript</b>			
Monday, February 21, 2000 <b>Lab: Introduction to JavaScript</b> Reading: <i>To be determined</i>	Tuesday, February 22, 2000 <b>Lab: Control Structures</b> Reading: <i>To be determined</i>	Wednesday, February 23, 2000 <b>Lab: Getting User Input</b> Reading: <i>To be determined</i>	Friday, February 25, 2000 <b>Problems of Online Images</b> Reading: <i>None</i>
<b>Week 06: Computer Graphics</b>			
Monday, February 28, 2000 <b>Lab: Images</b> Reading: <i>Handout: Simple Image Processing</i>	Tuesday, February 29, 2000 <b>Image Compression</b> Reading: <i>Dewdney 47 (Storing Images)</i>	Wednesday, March 1, 2000 <b>Image Processing</b> Reading: <i>Dewdney 32 (The Fast Fourier Transform)</i>	Friday, March 3, 2000 Reading:
<b>Week 07: Miscellaneous</b>			
Monday, March 6, 2000 <b>The Game of Life</b> Reading: <i>Dewdney 44 (Cellular Automata)</i>	Tuesday, March 7, 2000 <b>Genetic Algorithms</b> Reading: <i>Dewdney 16 (Genetic Algorithms)</i>	Wednesday, March 8, 2000 <b>Review for Midterm Examination</b> Reading: <i>Review Handout</i>	Friday, March 10, 2000 <b>Midterm Examination</b> Reading: <i>Midterm Examination</i>
<b>Week 08: Reliability</b>			
Monday, March 13, 2000 <b>Some Background</b> Reading: <i>Forester&amp;Morrison 5 (Unreliable Computers)</i>	Tuesday, March 14, 2000 <b>Proving Programs Correct</b> Reading: <i>Dewdney 10 (Program Correctness)</i>	Wednesday, March 15, 2000 <b>The Costs of Reliability</b> Reading: <i>None</i>	Friday, March 17, 2000 <b>To Be Determined (Something Fun)</b> Reading: <i>To Be Determined</i>

<b>Week 09: Database Algorithms</b>			
Monday, April 3, 2000 <b>Introduction to Databases</b> Reading: <i>Dewdney 65 (Relational Data Bases)</i>	Tuesday, April 4, 2000 <b>Search Trees</b> Reading: <i>Dewdney 11 (Search Trees)</i>	Wednesday, April 5, 2000 <b>Hashing</b> Reading: <i>Dewdney 43 (Storage By Hashing)</i>	Friday, April 7, 2000 <b>Text Compression</b> Reading: <i>Dewdney 52 (Text Compression)</i>
<b>Week 10: Database Implications</b>			
Monday, April 10, 2000 <b>Text Indexing</b> Reading: <i>To Be Determined</i>	Tuesday, April 11, 2000 <b>Invasion of Privacy</b> Reading: <i>Forester&amp;Morrison 6 (The Invasion of Privacy)</i>	Wednesday, April 12, 2000 <b>Ownership of Data</b> Reading: <i>To Be Determined</i>	Friday, April 14, 2000 <b>Database Consistency</b> Reading: <i>To Be Determined</i>
<b>Week 11: What is Computable?</b>			
Monday, April 17, 2000 <b>Models of Computation</b> Reading: <i>Dewdney 2 (Finite Automata) 7 (The Chomsky Hierarchy) 66 (Church's Thesis)</i>	Tuesday, April 18, 2000 <b>Costs of Computing</b> Reading: <i>Dewdney 26 (Nondeterminism) 54 (NP-Complete Problems)</i>	Wednesday, April 19, 2000 <b>The Limits of Computing</b> Reading: <i>Dewdney 5 (Godel's Theorem) 59 (The Halting Problem)</i>	Friday, April 21, 2000 <b>Implications of Computing's Limits</b> Reading: <i>None</i>
<b>Week 12: Artificial Intelligence</b>			
Monday, April 24, 2000 <b>Introduction to Artificial Intelligence</b> Reading: <i>Forester&amp;Morrison 7 (Artificial Intelligence and Expert Systems)</i>	Tuesday, April 25, 2000 <b>Game Trees</b> Reading: <i>Dewdney 6 (Game Trees)</i>	Wednesday, April 26, 2000 <b>Neural Networks</b> Reading: <i>Dewdney 27 (Perceptrons) 36 (Neural Networks that Learn)</i>	Friday, April 28, 2000 <b>Expert Systems</b> Reading: <i>Dewdney 64 (Logic Programming)</i>
<b>Week 13: Computer Viruses</b>			
Monday, May 1, 2000 <b>Background</b> Reading: <i>Forester&amp;Morrison 4 (Hacking and Viruses)</i>	Tuesday, May 2, 2000 <b>Core Wars</b> Reading: <i>To Be Determined</i>	Wednesday, May 3, 2000 <b>The Technology of Viruses</b> Reading: <i>Dewdney 60 (Computer Viruses)</i>	Friday, May 5, 2000 <b>Viruses, Revisited</b> Reading: <i>None</i>
<b>Week 14: Intellectual Property</b>			
Monday, May 8, 2000 <b>Software Theft</b> Reading: <i>Forester&amp;Morrison 3 (Software Theft)</i>	Tuesday, May 9, 2000 <b>Ownership of Algorithms</b> Reading: <i>To Be Determined</i>	Wednesday, May 10, 2000 <b>Course Summary and Evaluation</b> Reading: <i>To Be Determined</i>	Friday, May 12, 2000 <b>Review for Final</b> Reading: <i>Handout: Final Review</i>

# Class 01: Introduction to the Course

Held Monday, January 24, 2000

## Overview

Today, we begin our consideration of computer science by considering the subject of computer science: What is it? What do computer scientists study? We also go over some administrative issues.

**Today's Question:** *What is computer science?*

**Next Question:** *What computing knowledge do you hope to get from this course?*

## Notes

- *Assignments:* (due tomorrow)
  - Read the introductory handout
  - Fill in the introductory survey
  - Scan through Getting started in the MathLAN. (It will make much more sense once you're in front of a computer.)
  - Send me an answer to "What computing knowledge do you hope to get from this course?"

## Contents

- Defining Computer Science
  - Algorithms
  - Intellectual Foundations
  - Social Issues
- Administrative Issues

## Summary

- Definitions
  - Computer Science
  - Algorithms
  - Social Issues
- Course Issues
- *Handouts:*
  - Getting Started in the MathLAN
  - Introductory materials

## Defining Computer Science

- I like to begin each class by considering our initial perceptions of the subject matter of the course. One way to do so is to elicit definitions.
- What is *computer science*?
  - Take a few minutes and write down a definition. I'll then ask you to read them aloud.
- Some typical definitions:
  - A fancy term for computer programming.
  - The science of computers or computing
- What is *science*?
  - A method of studying (typically studying the natural world) based on observation, postulation of hypotheses, and support of those hypotheses by experimentation.
  - ...
- What is *computing*?
  - The solution of problems by the application of a fixed method?
  - ...
- Does computing require a computer?
  - Arguably, no. We can apply instructions by hand even if we don't understand the logic behind them.
  - However, computers are particularly good at doing lots and lots of simple operations.

## Algorithms

- Many computer scientists consider computer science to be the study of *algorithms*, formal sets of instructions for completing tasks. (Note that our definition of computing emphasizes these sets of instructions.)
- Computer scientists write and analyze algorithms.
  - Some are implemented in physical devices (hardware).
  - Some are expressed in ways the hardware can run them (software).
- Computer scientists write languages in which to express algorithms.
- Computer scientists study whether or not algorithms can be written for all problems, and how efficient those algorithms can be.
- ...

## Intellectual Foundations

- Computer science draws upon a number of other fields for its intellectual foundations: science, mathematics, and engineering.
  - Increasingly, computer science is also drawing upon psychology.
- From mathematics, we get formal approaches to solving problems and verifying solutions.
- From science, we get experimental approaches to solving problems and evaluating solutions.
- From engineering, we get techniques for building and analyzing large systems.
- From psychology, we get techniques for understanding (and improving) the interaction between computers and people.

## **Social Issues**

- It is also clear that computers are having a significant impact upon modern society (particularly modern American society).
- Many computer scientists consider it important to consider the ramifications of their work and fields.
- In this course, we will try to balance algorithmic and social issues as we consider a variety of topics.
- This is my three-course semester, and two of those courses are new to Grinnell. Hence, expect me to be stressed out and behind in work all semester.

## **Administrative Issues**

- Please refer to the course web site and the introductory handout for details.
- Teaching philosophy: I support your learning
- Work: Many small assignments; nothing large
  - Do the reading in advance of each class
  - One-paragraph answer to a question
  - Read the risks digest
  - Three narratives
- Policies
  - Attendance: I expect you to attend every class. Let me know when you'll miss class and why.
  - Grading: I'm a hard grader. I don't grade everything.
  - Course web
  - Etc.
- Warnings
  - Computer science has a mathematical underpinning. Some of our readings will be mathematical. Try not to glaze over the math.
  - This is the first time this course has been taught. I expect that many things will change over the semester.
- The syllabus

## How to Use the Course Web

For a number of reasons, I have chosen to make many of the handouts for this course available only in electronic format on the World-Wide Web. I will not go over basic use of the Web, since you should know about it from other courses. You should make sure to ask me if you have any questions about using the World-Wide Web.

The course web can be found at [You may want to bookmark that page.](#)

A number of important pieces of information are in the course web, including assignments, readings, requirements, syllabus, and office hours. I assume that if I put information on the Web, you will (eventually) read it.

- At the bare minimum, you should read all the pieces of basic information about the course. Of particular interest is the syllabus, which lists all the readings. I will also hand out a copy of this information on the first day of class.
- I prepare a rough outline for each class. Most students find these useful, and you should feel free to refer to them before, during, and after class. Since this class is becoming more lab-based, it is likely that in laboratory sessions there will be more information in the outlines than I will cover in class.
- Each outline begins with some notes. You can find just the notes in a separate news page.

At the top and bottom of every page are a series of links to important components of the course web. These are

- Instructions. This set of instructions.
- Search. A simple search facility for the course web.
- Current. The outline of the current or next class. You may need to reload the page to get the appropriate version.
- News. The course news, taken from the outlines.
- Syllabus. The course syllabus.
- Glance. An abbreviated version of the syllabus.
- Links. A collection of links that you might find useful.
- Handouts. Handouts for the class.
- Stories. Short stories of computing and society. You will need to contribute your own this semester.
- Outlines. The outlines of classes that have been held. You can sometimes access other outlines through the syllabus.
- Labs. Laboratory assignments. When appropriate, we'll do labs in class (or as take-home assignments).
- Assignments. A list of the assignments for the class, accompanied by their due dates. (There may not be any assignments except those mentioned in the course front door.)
- Examples. A list of examples generated for this class.

# Administrative Information

## On Teaching, Learning, and Grading

- Introduction
- My Role
- Grading
- Your Role
- Lecturing
- Favoritism
- Summary

### Introduction

I like to begin each course with a metacommentary on teaching and learning. Why? Because I care about the learning process, because I seem to have a different teaching style and personality than some students expect, because I want you to think not just about *what* you are learning, but also *how* you are learning, and, unfortunately, because in one of the first two courses I taught at Grinnell some students were clearly dissatisfied with the way I teach. (As people are getting used to me and my teaching style, this seems to be less of a problem.)

From my perspective, you are here to learn and I am here to support that learning. What will you be learning? The subject matter of the course, certainly. However, I expect that (or hope that) you will also be discovering new ways to think and learn or sharpening existing skills. In terms of subject matter, I tend to care more about the processes and concepts that you learn than about the “basic facts”.

Learning is an interactive process. You learn by asking, discussing, and answering questions, by playing with ideas (in computer science, you also learn by playing with programs), and by working with others. I know from experience that computer science cannot be learned passively: you need to experiment with ideas (in your head, on paper, or on the computer) in order to fully grasp these ideas.

### My Role

How do I try to support this learning? In a number of ways.

I *assign readings* to give you a basis for understanding the subject matter. Sometimes these readings will be from the textbook, sometimes I will distribute appropriate supplements.

I *lecture*, *lead discussions*, and *conduct recitations* on the topics of the course. Sometimes these will be based on readings and assignments, sometimes they will vary significantly from your readings. Why? Because I feel it wastes your time and mine to simply reiterate the readings. If you let me know that you’re confused about a reading, I will spend time going over that reading (either in person or in class).

To stimulate discussion and thinking, I regularly *call on students* in class. I know that not all of you are comfortable answering questions publically, but I strongly believe that you need to try. Please feel free to say “I’m not sure” when I call on you. At times, I’ll step through the class, asking each student in turn. At others, I’ll call on you individually. I tend to call more on students I interact with regularly.

I *assign work* because I find that most people learn by grounding concepts in particular exercises that allow them to better explore the details and implications of those concepts. I expect you to turn in work on the day it is due and will impose severe penalties on late assignments (including refusing to accept some late assignments).

Some of my assignments may involve *public presentation* of your work. Sometimes, the best way to learn a topic is to have to discuss it or present it to someone else. In addition, I've found that many students need some work on their presentation skills. Most often, presentations will be of papers that you've read.

**In general, I expect you to spend about ten hours per week on this class outside of class time.** If you find that you are spending more than that, let me know and I'll try to reduce the workload.

I *grade assignments* to help you identify some areas for improvement. Note that I believe that you learn more from doing an assignment than from receiving a grade on that assignment. This means that you may not receive a grade or comments on all your assignments. I will tell you when an assignment won't be graded, but not until after you hand it in. I will do my best to be prompt about returning grades on assignments. At times, I will use a grader to help speed the process.

I *give examinations* because I find that many students only attempt to master a concept when preparing for an exam. Because I care more about processes and concepts than about facts, I almost always give open-book examinations. Because I do not feel that time limits are helpful, this semester I will give you only take-home exams.

I *give quizzes* to ensure that you are doing the reading and that you are understanding what I expect you to understand from the readings and assignments. At times, I will give quizzes to help illustrate a particular point. This semester, all of my quizzes can only affect you positively: good work on quizzes will lead to extra credit.

I *build course webs* to organize my thoughts, to give you a resource for learning, and to help those of you who need to work on your note-taking skills. I do my best to make my notes for each lecture available on the Web, in outline format. In general, these notes will be available approximately five minutes before class. Warning: these are rough notes of what I expect to talk about; the actual class may not follow the notes. I will also attempt to update the notes after each class.

I *make myself available* to discuss problems and questions because I know that some of you will need personal attention. In general, if I'm in my office you should feel free to stop in. Most of the time, I'll be willing to help. Once in a while, I'll be working on a project and will ask you to come back later. Students always have first priority during office hours. You should also feel free to send me electronic mail, which I read regularly, and to call me. This semester, I am on partial parental leave, which means that I will be less available than normal. In particular, I will not be in the office on Tuesdays and Thursdays. Feel free to give me a call at home on those days, but understand that I may be busy.

At times, I *survey* my students to better understand how the class is going. Because I do research on the effects of computers on learning, I sometimes give surveys to gather data.

## Grading

At the same time that you learn and I try to help you learn, Grinnell and the larger community expect me to assign a grade to your work in the class. I base grades on a number of components, but primarily on *assignments, examinations, and involvement in classroom discussions*.

Because I understand that not everyone gets everything right the first time, I will occasionally allow you to *substitute* an extra assignment for one that you did poorly on. Unfortunately, the time pressures of the semester are significant enough that I will not be able to permit you to make up assignments except through this mechanism.

In computer science, it is often possible to do the same problem in multiple ways. Hence, I typically reserve class days to discuss particularly significant assignments. This semester, each exam will be followed by a day of discussion relating to the exam. We may also take time from some classes to discuss particulars of assignments.

I will admit to a fairly strict grading scale. Grinnell notes that A and A- represent exceptional work. To me, “exceptional” means going beyond “solid”, correct work. Exceptional work entails doing more than is assigned or doing what is assigned particularly elegantly. Work limited to mastery of the core materials is B-level work. To help you demonstrate exceptional understanding, I will occasionally suggest *extra credit work* (although truly exceptional students will often suggest such work on their own).

To help eliminate biases, I typically use a numerical grading scale. 94-100 is an A, 90-93 is an A-, 87-89 is a B+, 84-86 is a B, and so on and so forth.

## Your Role

How should you participate as a member of my class? (Or, how do you do well in my class?) By being an active participant in your own learning. In part, this means doing all the work for the class. It also means a number of other things.

**Come talk to me** when you have questions or comments about subject matter, workload, or how the course is going in general. I will also set up an anonymous comment page for those who are uncomfortable talking to me directly.

**Do the readings** in advance of each class period and come prepared with a list of things that you don't understand. I will try to spend time at the beginning of each class session answering these questions or will restructure the lecture to accommodate them.

**Ask and answer questions and make comments** during class periods. I consider active participation during class a particularly important part of the learning process.

**Begin your assignments early.** Students who begin assignments early have more opportunities to ask for help, to make sure that the assignment gets completed, and to sleep at night. Such students also do better in general.

## **Lecturing**

I seem to have a different “lecturing” style than some students expect. As I mentioned earlier, I don’t think it is the purpose of lecture to reiterate the readings. I do, however, think lecture and readings can provide alternate perspectives on the subject matter. At times, I will also discuss issues not covered in any readings.

I see no point in going on with a lecture or example if many students don’t understand what’s going on. You are the first line of defense: stop me when you are confused. In addition, I will occasionally stop the class and ask for a show of hands to see who is confused. Don’t be embarrassed to raise your hand; if you are confused, it is likely that someone else is also confused. I realize that this show of hands leads to some “pressure for understanding”. However, you won’t get much out of a class if you’re confused (and therefore just copying down what I’m writing without thinking about it).

I deem it important for students to be active participants in lecture. This means that I will often ask you to help develop algorithms, solve problems, and even critique each other’s answers. If I call on you and you’re not sure of an answers, feel free to say “I don’t know” or to venture a guess. I consider it very important for all of us to see the problem solving process, warts and all. Note that I often generate examples of discussion “on the fly” so that we can all be involved in the problem solving or development process.

## **Favoritism**

For various reasons, I often get to know some students better than others. I tend to call on the students I know better, and sometimes respond slightly better to their questions because I have more context for the questions. I’m happy to make all of you “favorite” students. If you come to see me regularly and work enthusiastically on the material, you’ll probably end up being a student I know well.

## **Summary**

As the prior discussion suggests, I expect a great deal from my students. I also use many different strategies to get the best out of you. Feel free to discuss any of this with me (anything from concerns about this perspective to suggestions on improving teaching and learning).

## **Academic Honesty**

I expect you to follow the highest principles of academic honesty. Among other things, this means that any work you turn in should be your own or should have the work of others clearly documented. When you explicitly work as part of a group or team, you need not identify the work of each individual.

You should never “give away” answers to homework assignments or examinations. You may, however, work together in developing answers to most homework assignments. Except as specified on individual assignments, each student should develop his or her own final version of the assignment. On written assignments, each student should write up an individual version of the assignment and cite the discussion. On non-group programming assignments, each student should do his or her own programming, although students may help each other with design and debugging.

When working on examinations, you should not use other students as resources.

If you have a question as to whether a particular action may violate academic standards, please discuss it with me (preferably before you undertake that action).

## **Citing Program Code**

Note that computer programming shares with normal writing a need to cite work taken from elsewhere. It is certainly acceptable practice to borrow other code for your assignments. However, you must cite any code that you use from elsewhere. Each piece of code you take from elsewhere must include a comment that specifies:

- the author of the original code;
- the date the original code was written and the version of the code (if available);
- the date you incorporated the code into your program;
- a summary of the modifications (if any) you made to the code;
- instructions for getting the original code.

This applies not only to the code you get from the Web and elsewhere; it also applies to code you get from me and from the textbook.

You do not need to cite the classes and libraries you use, as the command to include classes and libraries within a program provides sufficient citation.

## **Disabilities**

I encourage those of you with disabilities, particularly “hidden disabilities” such as learning disabilities, to come see me about the accommodations that I can make to make your learning easier. If you have not already done so, you should also discuss your disability with academic advising. If you think you may have an undocumented learning disability, please speak to me and to academic advising.

In my experience, some learning difficulties can make computer science more difficult because of computers’ emphasis on small details. I also know that many of my favorite and best students have some learning disability, and have certainly succeeded. We’ll all do better if you talk to me about disabilities early.

Note that I generally feel that the “accommodations” that we are asked to make for those with learning disabilities are often appropriate for all students. Hence, I rarely give timed exams and I typically allow students to use computers during exams.