CS 135 Survival Guide

Why a survival guide?

CS 135 is designed to be well within the reach of all university students. There is no material in CS 135 that is beyond your abilities, and there is no reason that you should not succeed in it. However, the nature of computer science can make any introductory course a challenge to those who do not develop the right work habits.

This survival guide is a collection of guidelines, tips, suggestions, and background information to help you make the best of your opportunities. It is based on the experience of first-year students facing their first university course in computer science.

Course philosophy

We designed CS 135 for students in diverse programs, keeping in mind the large range of possible ways in which computer science might be integrated into future studies. By choosing to focus on fundamental concepts and study them in depth, we are giving students transferable skills on which to build (rather then providing a shallow overview of the breadth of the field of computer science).

The material in CS 135 lends itself very nicely to being built up in successive layers, where a new layer is added only when the previous one is completely explained and understood. One of our goals is to progress through the material in small steps rather than huge leaps.

When you use a Web browser, e-mailer, or other modern computer application, there's a lot going on behind the scenes. One of our goals is to have as little "magic" as possible. You might hear us saying "Know your tools." That means understanding not only what the tools do, but how they do it. In many cases, we will show how to implement built-in features of Racket, rather than just describe them.

Each component of the course has a role to play: lectures, the textbook and assignments teach you the material and give you a chance to practice; assignments and midterms provide diagnostics, giving you feedback on whether or not you are on the right track; and assignments, midterms, and the final are used for assessment (computing your mark in the course).

Doing the assignments is the key to doing well in the course. The process of arriving at the answer is more important than the answer itself.

Lectures illustrate concepts and techniques; assignments give you practice in those concepts and techniques, strengthening your existing skills and teaching you new ones. Most of the learning in CS 135 takes place through your working through problems on your own. In many cases the final answer isn't particularly relevant; it's going through the process of arriving at the final answer that's important. Doing the assignments is the key to doing well in the course. Skipping assignments would be like trying to learn to play a musical instrument without practicing, attempting to master

a language without speaking it, or claiming to be an outstanding artist or athlete based solely on theoretical understanding of the underlying principles of the field.

In previous offerings of first-year courses, we have discovered that the number of assignment questions submitted was a better predictor of final grades than were high school marks. Some students make the mistake of thinking that these components aren't important because they only account for 20% of the final grade, or that they can skip one assignment because it's worth so little. But an assignment question worth less than 1% of the final grade could provide important insight into a concept tested by a final exam question weighted more heavily.

You must earn an overall passing grade in the assignments in order to pass the course.

If assignments are so important, why aren't they weighted as a greater percentage of the final grade? Your final grade is supposed to be an indication of how well you have mastered course material. That's assessment, which occurs after learning. Lectures are part of the learning process, so you don't receive marks for attendance. If we took the assignments out of the final grade computation, you'd be tempted not to do them, and put yourself in serious danger of failing the final exam. So we make them worth a little bit to encourage you to do what you should be doing for your own good anyway. In fact, you must earn an overall passing grade in the assignments in order to pass the course.

Lectures

We've tried to create a lightweight implementation of the basic course outline. Instead of having a fixed set of slides for each lecture, we've grouped slides into thematic units we call lecture modules, with only a rough idea of how long we will take to cover each one. That way, if you ask a lot of questions, we don't have to rush over subsequent material to catch up with a ticking clock.

The instructors work from the same set of lecture notes, though each will say different things, and possibly write different examples on the board (or overhead projector, or data projector). If you have to miss your lecture, it is better to attend another lecture than not attend any that day. However you should be aware that as the sections may drift a little out of sync, you may still end up missing some material. In addition, you should be sure that you are not taking a seat from someone who is legitimately enrolled in that section.

Deadlines are firm.

For possibly the first time in your life, no one is forcing you to go to lectures. You are free to not attend. This honour system is built on the assumption that this freedom will be exercised responsibly. Other aspects of this assumption include the assumption that students will complete the required work on time, as deadlines are firm and no "bonus assignments" are handed out to bolster marks.

You may be tempted to skip lecture because the text of all lecture slides is available on the web; surely you can do just as well reading them? If we thought that, we wouldn't be lecturing. The slides are like basic musical chords over which we improvise a melody; the chords alone do not make up the whole song. Not everything is written down on the slides. It's not that we've

deliberately left things out; it's just that we don't design the slides to be the sole source of learning, but rather as an aid in lecturing.

The handouts contain the text of the slides so that you don't need to scribble madly while we display them. That frees up your time so that you can take proper notes on what we say in class. We've seen students write down only what we write on the board, or not write anything down at all, figuring that the slides do all their work for them. That's a mistake; you can't possibly remember all of what is said in all of the lectures you will attend during a term. Taking notes helps fix information in your long-term memory, keeps you active, and allows you to preserve insights that you gain during the course of a lecture.

You can't remember all the important stuff in a lecture; take notes.

Going to lectures reserves three hours of dedicated time each week during which you are thinking about course material, with the aid of someone who understands it thoroughly. It's a rare student who is disciplined enough to skip lectures and adequately substitute for that experience. Of course, not every minute of lectures is "quality time", but you won't know what you're missing if you don't attend. Take something discreet to do (like your math homework or a good novel) during the stretches where there's an extra example of something you already understand, and make sure you keep one eye and ear on what's going on. You're sharing space, so don't do anything that might annoy or distract your classmates: don't let your cell phone ring, don't rustle food wrappers loudly, and don't play video games on your laptop.

The textbook

CS 135 wouldn't exist if it weren't for our textbook, "How to Design Programs". CS is not a field blessed with an abundance of good textbooks, due to its very short history and the ever-changing nature of the curriculum. This is one of the better textbooks. The lectures are designed on the assumption that you'll do the assigned readings—before lecture if possible, but certainly before you attempt to do the corresponding assignment. This is not a textbook to be used for reference, to be looked at only when you run into trouble on an assignment. That will only increase the amount of time you'll need to get your work done. (We'll say more about doing assignments below.)

On occasion, we will use the same examples in lectures as in the textbook, illustrating different aspects. More often, we will use different examples, and at times, we will "parallel" the book by covering the same concepts using different language and a different approach. Sometimes, we will go more in depth than the textbook (it assumes eighth-grade knowledge of mathematics, and we can expect better than that of you) and sometimes we will skim a topic, leaving the details to the readings. But the textbook always informs and directs our efforts.

The textbook and lecture slides may contain conflicting information, especially with regards to the design recipe. The lecture slides always take precedence over the textbook. You should follow the design recipe as taught in lectures, and not the textbook.

The textbook is available on-line at `http://htdp.org`. If you want, you can purchase a physical copy at the UW Bookstore in SCH or at the Feds Used Books store in SLC.

Assignments

It is on assignments that poor work habits may be most obvious, since programming, approached incorrectly, can quickly become a timesink. It's possible to spend far too much time on CS assignments, because it's easy to believe that you're really close to a solution and the next change you make to your program will have it running perfectly. Hours go by before you realize it. Don't waste your time grinding away at a question; if you feel you're not making progress, go on to another question, or even switch to working on another course, and come back later. We've had students say, "I can't find an example in the textbook that looks like this assignment question". We try not to ask assignment questions that look like examples, because we want you to be able to write programs from scratch, not just modify examples that you don't fully understand.

Use the design recipe.

Some students try to do CS assignments by looking at the assignment for the first time while sitting in front of their computer. They read the first question and immediately start to write a program. We'd like to suggest a more productive approach. Read the assignment away from your computer, and think about how you might solve the questions. The lectures and the textbook teach something called the **design recipe**. This is a process to go through in writing a Racket function or set of functions to solve a problem. The authors of the textbook came up with the design recipe by observing students in the lab and noticing when and how they ran into difficulties, and what worked to help overcome their difficulties. We encourage you to use the design recipe by assigning marks to its various components (since they help us to see that you understand what you are doing). You can, of course, get your program running and then put in all the components worth marks afterwards. But following the process properly will save you time and reduce frustration. In CS 135, the programs you have to write are usually quite small; most of your time should be spent on thought.

Use the Basic Tests.

The design recipe may include templates, which are function skeletons derived from data definitions. Some students in previous offerings of CS 135 have reported that the assignments took half as much time when they used the templates as we suggest. We also provide **Basic Tests**. You must use the Basic Tests to prevent losing marks when our automarking software can't find your answer because you spelled the name of your function incorrectly.

You may use your own computer, or you may use the computers in the public Mac labs in the Math & Computer building. Details on how to copy your files to/from the Macs, how to electronically submit your work, and some details of the program style that we expect can be found in the **submission and style guide**, also included here in the course handouts.

We have scheduled assignments (roughly one per week) so that the work is spread out evenly across the term. We recommend that you start each assignment shortly after finishing the previous assignment, putting in a solid but not excessive amount of time. This allows you to have time to set it aside and come back to it later. Starting early allows you to discover difficulties and get help, and you will find that when you return to questions fresh, you will see ways to solve problems that you couldn't see before. Although this advice holds for any course, it is particularly important for computer science.

You should try to complete the whole assignment, but if it turns out that you don't have time

to do it all, submit what you have, even if it is incomplete. You will still get written feedback from the markers, and you may get some marks as well. When you skip an assignment, you are likely to find the next one more difficult. Do your best to avoid skipping assignments, and make sure you have a sense of how to do each question, even if you don't submit each one. Finally, when you get back your assignments, don't just glance at the mark and toss it aside. Learn from your mistakes.

Look over your marked assignments; learn from your mistakes.

Model solutions will be posted on the course web site shortly after each assignment is due. In some cases several alternate solutions will be presented. Study these to make sure you understand the material; this will pay off on later assignments and on exams. This is true even for students who receive perfect marks, as even a solution that has no obvious flaw might be able to be improved in elegance or other attributes. If nothing else, knowing how to find a shorter solution will be of great benefit on exams.

The midterm exams

The midterm exams have a diagnostic purpose: they give you feedback on what you thought you understood (but didn't) and what you thought you didn't understand (but did). Having a chance to take an exam before the final also gives you an indication of what that experience might be like. As with assignments, if we don't make them worth some percentage of the final grade, students tend not to take them seriously, and so their value is lost. So midterms account for a modest fraction of the final grade.

Midterm questions will resemble assignment questions, except that we can't ask you questions that will take too much time, either to discover the answer or to write it down. You also won't have a computer to use during the midterm (or on the final exam), meaning we can't ask you questions for which you need to go through a process of discovery and refinement (as we can on assignments). We won't ask you to write down memorized definitions. Obviously, you need to know the syntax of Racket and the meanings of terms and phrases that we use. But we will test that understanding by asking questions where you need to use this material, not just repeat it from memory.

The course does change from term to term.

The best way to do well on the midterms is to learn the material as it's presented in class and exercised in tutorials and on assignments. It helps to plan for exam study as part of your weekly schedule. Start preparing review sheets as you learn material rather than waiting until right before an exam. During the midterm exam period, there is a temptation to neglect all courses except the one with the next exam; if you do this, you may find yourself behind and unable to catch up. Your studying should be active. Find questions in the textbook not already assigned and write out complete answers. Course personnel will be happy to answer any questions you encounter along the way.

We recommend not using midterms from past years as study guides. Each term, we create midterms from scratch, and the course does change from term to term. Past midterms may give you a misleading sense of what this term's tests will be like. As with assignments, we won't ask

you to do things that are just minor tweaks on lecture examples; we try to design questions that test deeper understanding. We also recommend not cramming at the last minute. This type of study results in shallow, easily forgotten understanding of a sort inappropriate to CS 135. Cramming works best when you have to repeat definitions and do things nearly identical to what you've already done, and we're not going to ask you exam questions like that.

1A midterm marks are a shock to some students, because they're the first confirmation that grades tend to be lower in university than in high school. We don't plan it that way, but we're probably asking more of you than your high school did, so it makes sense. We don't have a predetermined average or failure rate in mind, and we will not adjust the marks with a "bell curve" or any other method, unless we feel that a test was somehow flawed. We try to design tests that are good indicators of the necessary breadth and depth of knowledge. Typical midterm averages in first-year CS courses are between 65% and 75%.

Don't let a low mark throw you. Examine your study and work habits, with the advice of an instructor or tutor, and see what you can do to improve the results of the next test. As a significant fraction of your mark is based on the final, you have adequate opportunity to improve your performance.

Don't let a high mark lead you to become complacent and slack off. CS 135 gets harder as the term progresses (most courses do, since later material can build on earlier material). But the course is designed so that students can do well if they approach it correctly.

The final exam

Start studying early, study actively, and don't cram.

Most of the comments we made above about the midterms also apply to the final exam, but there are some key differences. The final exam isn't diagnostic, but straight assessment. For that reason, it makes up the most significant component of your final mark, which will be viewed by observers as an indication of how well you have mastered course material. Also, you don't get the final exam back, so there is no chance to learn from your mistakes.

You should approach the final in the same fashion that we recommended you approach the midterms: start studying early, study actively, and don't cram.

In order to pass the course, you must pass the weighted exam mark (the sum of each exam percentage multipled by its weight, all divided through by the total exam weight).

Final grades

You'd think that there wouldn't be much to do once the final exam is over: enter the grades into the spreadsheet, and submit the final marks. But we have to make sure that the marks reflect what we know of the students with whom we've been in contact. We go over the spreadsheet looking for anomalies (for example, students who have good marks up until the final exam). We also go over the exams of all students with computed final marks between 46 (sometimes lower) and 49,

looking for evidence that they deserve to pass. The students who end up failing (and we hope that this number is as small as possible) are those for whom we cannot find any evidence that they should continue with a successor course. In previous years, these were students who did poorly on the midterm exams and didn't take steps to improve their habits, or those who did okay on the first midterm, worse on the second, and submitted few or no assignment questions in the second half of the course.

Since entry to CS 136 requires a mark of 60 or better, we will review the situation for students with computed final marks between 56 (sometimes lower) and 59 as well.

In order to pass the course, you must pass the weighted exam mark.

The exam marks are a crucial part of your mark. As noted above, in order to pass the course, you must pass the weighted exam mark (the sum of each exam percentage multiplied by its weight, all divided through by the total exam weight).

Getting help and avoiding trouble

Sources of assistance

We encourage you to ask course personnel for help. The tutors have regular office hours, as do each of the instructors; the tutors answer e-mail sent to the course account, and the instructors will answer personal e-mail. Generally, office hours are not used enough, except perhaps right before an assignment is due. Note that course personnel tend to be available during the day; if you like to work late at night, that's okay, but be warned that help is generally not available at that time.

Getting help from fellow students can be problematic. While we encourage you to talk about ideas learned in lecture or from the book, you should avoid asking others about specific questions on assignments. You might find yourself accused of plagiarism, which we discuss in the next section.

Plagiarism

Plagiarism is the act of representing someone else's work as your own. Some students do this deliberately, believing that getting a good mark on an assignment by copying from a friend is more important than doing the work themselves to learn the material. This will hurt them on the midterms and final exam, which largely determine their final grade. But it also hurts other students, because the plagiarist has earned marks they do not deserve, and that will be taken as evidence of their having demonstrated skills that they do not really possess. When they turn out to be incompetent, in later classes or in the workplace, it reflects badly on their classmates; even those who earned their marks fairly will come under suspicion.

For this reason, UW takes a firm stand against plagiarism. The course Web page contains a pointer to UW's Student Academic Discipline Policy. The standard penalty for plagiarism on assignments in all CS courses is a zero on the assignment, a deduction of at least 5% in the final mark, and a letter to the Associate Dean for Undergraduate Affairs to be placed in your file; penalties are

more severe for cheating on exams, as well as for repeat offenses in this or other courses. We have access to software that analyzes assignment submissions for similarities and isn't fooled by simple tricks like renaming variables.

Do not take notes during discussions with other students.

Some students inadvertently stray into plagiarism by not being careful when they talk with others. Confine such discussions to high-level overviews of general concepts, not details of how to solve specific questions. Do not take notes during such discussions, and when you write or type up the ideas, use your own words and phrases. A good way to make sure that your write-up is different from others' is to leave some time between the discussion and the write-up (planning for this has the nice side benefit of getting you to start on your work earlier). Don't look at someone else's programs written for an assignment, or show your programs to someone else. Don't search on the Web or in books other than the textbook for answers to assignment questions, or even for hints.

The penalty is applied equally to those who use answers supplied by others and to those who supply the answers. "Helping" a friend by showing them your assignment is not only unfair to others and a violation of academic rules, but it will also result in your being penalized along with your friend. Don't do it.

Summary

Surviving CS 135 isn't like running an obstacle course; it's more a matter of applying common sense. Look for the reasons behind everything you're asked to do in the course—if you can't figure them out, ask an instructor or tutor, who should be able to answer your questions—and keep those reasons foremost in your mind as you do your work. It's important to us that you not only get through the course, but that you enjoy yourself and see the material as interesting and valuable.