Department of Computer Science
University of Toronto

Computer Science 148/A58
Introduction to Computer Science

Administrative Information
September 2002

Edited by Diane Horton

Contents

1 Introduction

Welcome to CSC148/A58!

This handout contains important information about the course, including some rules you need to know about. Please read it carefully. Term-specific and campus-specific information, such as course office hours, assignment due dates, and where to hand in assignments, is available on the course Information Sheet and the course website.

2 Web site and newsgroup

Lots of information about the course will be available through the CSC148/A58 web page. Important course announcements are made there throughout the term, and **you are responsible for reading them regularly.**

The course newsgroup is `ut.cdf.csc148h`. It provides a forum for discussion among students in the class. You are not responsible for reading the newsgroup, but you may find it helpful. If you have trouble reading the newsgroup from your home computer, it may be because your internet provider does not import it. You can certainly read it from the web browser in our computer lab.

3 Policy on electronic correspondence

Please follow these guidelines when sending email to us or posting to the newsgroup:

- **No mime:** Please do not send messages that include HTML or MIME, since plain ASCII text is the standard for communication by e-mail and newsgroup postings. See

  `http://www.expita.com/nomime.html`

  to learn why and to find out how to turn off HTML and MIME.

- **Don't post solutions:** Never post to the course newsgroup your solution to an assignment, or even your idea of the solution to an assignment, or even one small part of a program that is part of your solution to an assignment. *If you do this, it will be treated as a case of plagiarism with all the consequences that this entails.*

- **Check the webpage first:** Before asking a question by e-mail or in the newsgroup, please check the course webpage carefully — the answer to your question may already be there.

- **Check the documentation first:** If your question is not about the course material but about the computer system, then please check the system documentation first. It contains more detailed and accurate answers than we can provide you with. Note that we cannot answer questions specific to your computer at home, because we are not necessarily familiar with the operating system you're using.

- **Use a descriptive subject line:** Always use a descriptive subject for your e-mail, and always include the course identifier (csc148 or A58) in your subject. Your instructor probably teaches more than one course during each term, and it's not always obvious what course your question is about. Also, subjects like "A question" are not descriptive enough. At least, say what the question is about (lecture material, assignment, test, etc.).

Don't let these guidelines discourage you from asking questions! They are only meant to help you find the information you need quickly and accurately, and to leave us enough time to answer everyone's questions.

4 Getting help in office hours

Office hours are listed on the course Information Sheet. You are encouraged to use these office hours regularly to get help with the course. Often it is the best students who come to office hours. Some students never do, and end up struggling unnecessarily.

If you can, come to the office hours with a specific problem or question. But if you are too confused to find something specific to ask, then it is even more important that you come talk to us. We can help identify what it is that's causing your confusion, and then work with you to solve the problem.

Most students come to office hours with questions about the concepts in the course, but any questions concerning the course are welcome.

Bringing a bug to office hours

If you want help with a bug, you'll need to do some preparation so that we can help you fix it quickly.

Before bringing your bug to office hours, try to fix it yourself. Debugging is an essential skill that you'll be honing through extensive practice `:-)` over the next few years. When you have a logical error, these are some general steps you can take to fix it:

1. Use every clue that Java gives you (the error message, information showing which line crashed and which method calls lead to it), together with your powers of deduction to narrow your focus to a small piece of code that is likely the source of your bug. This should be at most about 20 lines.

2. Then use the debugger, or just output statements, to see what the values of the relevant variables are just before and just after that small piece of code. If the before values are sensible and the after values aren't, then you have successfully isolated the bug.

3. Once you have isolated the bug, get out some paper and *trace the offending code by hand.* Your trace will show you what's happening (and why), which will not be what you wanted to happen (hence the bug!). Now you know what needs to be fixed. If your trace indicates that the code *does* do what you wanted (and yet step 2 tells you that it doesn't), then there is an error in your trace. Do it again, being even more detailed and accurate.

(If you don't know how to follow these steps, be sure to come to office hours and ask for help.)

Almost all of the time, these steps will show you the bug and you won't even need to ask us for help with it. But if they don't, perhaps because you have a misconception that prevents you from doing a correct trace, then ask us for help. In order to help you, we'll need you to bring along:

- A completely up-to-date printout of your code. If it has hand-written changes, we'll ask you to get an up-to-date printout.

- An explanation of exactly what small piece of the program it is that you've narrowed the bug down to.

- Printed input and output from a test run showing the state of affairs before and after the offending piece of code is executed.

- Your careful and detailed hand trace of the offending piece of code. If it looks like a bunch of scribbles, we'll ask you to do it more carefully.

5 Keys to a good assignment

Although each assignment will have different requirements, below are some general guidelines to help you understand what sorts of things we require in a good assignment. You should also look at the sample assignment and its solution available on the course web site.

Simply writing a working program will not guarantee a good mark, possibly not even a pass. To earn a good mark on the correctness of your program, you must *demonstrate* that it works by performing and documenting thorough testing. Without this, the marker can only assume the program doesn't work properly, and you will earn a very low mark for correctness. In addition, to get a good mark on a program it must be well designed and well documented. You may also be required to hand in written answers to questions. These will be marked for quality of writing as well as for their content.

It is important that you learn to go beyond just writing a program that you believe works; the skills required to do this are essential for a successful computer professional.

Here are some of the things we're looking for when we mark assignments:

Correctness

The program should work correctly for all valid cases, including all "normal" input cases and all boundary cases. (A "boundary case" is valid but is near the boundaries of what is valid.) Your program should give appropriate error messages for invalid input data that you were asked to handle. If you are not sure what cases you are expected to handle, ask. Any output that you design should be self-explanatory and formatted so that it is easy to read.

Testing and explanation of testing strategy

Choose a set of test cases that systematically and thoroughly puts the program through its paces. Produce test output that is easy to digest. In a separate document, explain why all of your test cases, taken together, provide convincing evidence that the program is correct. For more guidance, read:

- The document "Software Testing," available in the online course handbook at
  `http://www.cs.toronto.edu/~dianeh/148/handbook/`

Class design

The program should be divided into appropriate classes. Inheritance should be used to express relationships among the classes. Access modifiers (`public, private`, etc.) should be used on classes, data members, and methods to enforce information hiding. Classes should be organized into packages (if you have studied packages).

Lower-level programming style

Code should be indented in such a way that its structure is visually apparent. Code should not be unnecessarily complicated. Constants should be used rather than "magic numbers". Identifier names (the names of packages, classes, interfaces, variables, and methods) should indicate meaning clearly. Variable and constant declarations should be localized appropriately. Data structures should be well chosen. The program should not waste time or space. No method should be more than about 40 lines long, including comments. Parameters should be used appropriately for communication between methods. If your programming style is good, it will be easy to modify the program or to reuse individual classes or packages in other programs. For more information, read:

- *The Practise of Programming*, by Kernighan and Pike; Addison-Wesley, 1999.

Comments

Comments should be grammatical, precise, and concise. They should be helpful, rather than being simple repetitions of the code. Every method should include a paragraph-long comment explaining what it does under all circumstances, in terms of the parameters. Any preconditions on successfully using the method should be given. Comments should also summarize what's happening in each chunk of code within a method, and explain any particularly tricky sections of code. For more information, read:

- The section of the lecture notes on "Design by Contract."

Bugs and missing features

For most assignments, you will be required to include a (possibly very brief) written report on any bugs or missing features. It's not the end of the world if your testing reveals a bug you couldn't fix or a feature you didn't complete, but you must be honest about these problems, and you must carefully document exactly what does and doesn't work. If you know of no bugs or missing features, you should say so.

Quality of writing

Written components of an assignment include the comments in your code, your description of your testing strategy, and your answers to any questions. Proofread carefully to be sure that your grammar, spelling and punctuation are correct. Try to be concise. Above all, your meaning should be clear, and your arguments well made. For advice and references on how to improve your writing, see:

- Section 6 of this document.

Professional presentation

Written work should be typed. Always include the following information on every document you hand in: your name and student number, your section, your professor's name, and your TA's name. Make sure that all items are well organized and clearly labelled.

6 The importance of good writing

A high level of English language proficiency is essential to success as a computer scientist.

> A typical commercial software project involves creating more than 20 kinds of paper documents on such items as requirements and functional, logic, and data specifications. For civilian projects, at least 100 English words are produced for every source code statement in the software. ... Many new software professionals are surprised when they spend more time producing words than code.[1]

[1]Capers Jones, "Gaps in programming education", *IEEE Computer*, April 1995, page 71.

If you would like help with writing, speak to your professor or take advantage of the excellent writing resources available on campus. See

http://www.library.utoronto.ca/www/writing

for information about writing centres on all three campuses, as well extensive online handouts on writing and links to many useful online resources.

The following book may also be helpful:

- *Writing for Computer Science: The Art of Effective Communication*, by Zobel; Springer, 1997.

7 Assignment due times and late policy ⇐ READ THIS

Assignments are due at the time and place stated on the assignment. **The due time is strict.** An assignment that is handed in 15 minutes later is considered to be late.

The late policy is described on the course Information Sheet. **The late policy is strictly enforced**, so be sure that you are familiar with it. Do not assume that you can always hand in an assignment later for a reduced mark.

Each assignment will have very specific requirements for where and how you are to hand it in as well. For instance, you will often be required to hand in your code electronically, using particular file names and directories. You may be told that your assignment will not be graded if you do not follow the guidelines. (See the course web site and your assignment handout for such details.) These requirements are very important due to the size of the course, and they will also be strictly enforced.

8 If you can't finish an assignment

The rules about lateness are strict partly because you simply cannot afford to fall behind in this course. If you think you won't be able to finish an assignment on time, it's usually better to hand in a partial assignment on time than to hand in a complete one late.

For an assignment with only written questions and no programming, it's obvious how to do this: just hand in the questions you were able to answer. But you can also hand in a partial solution to a programming assignment. Now if you just hand in a program that's full of bugs and won't run at all, then you will get a very low mark. Instead, define a simplified version of the problem and create a working program for *that*, with good documentation (including an explanation of the simplifications that you made), thorough testing, etc. You should also try to do as much of the non-programming parts of the assignment as possible — for example, answer any questions that we assign. This sort of assignment solution may well get a reasonable mark, and it will allow you to avoid delaying your other course work to put in more time on your 148/A58 assignment.

You will probably need to decide on your strategy at least a half day before the assignment is due, so that if you choose to hand in a simplified solution on time, you will have enough time to put it together.

9 If you become ill

Students may request special consideration in the event of illness or other catastrophe. This is a rare circumstance, and supporting documentation (e.g., medical certificate) must be presented.

If you feel you qualify for special consideration on an assignment, contact your professor immediately. We will ask you to fill out the "Request for Special Consideration" form on on the course web site, and to give both it and your documentation to your professor as soon as you recover.

Medical notes must be written on the standard University of Toronto Student Medical Certificate.

If you must miss a test during the term, contact your professor immediately.

10 Some more advice

The assignments in this course take quite a lot of time to complete, both time on the computer and time away from it designing your programs, planning your testing, answering written questions, etc. Do not leave assignments until the last minute! It is to your advantage to start working on your assignments as early as possible. This will give you more time to get some help if you have questions or difficulties.

A common strategic error made by CSC148/A58 students is to focus on the programming aspect of their assignments and to leave everything else (*e.g.*, comments, a testing strategy) until the program is done. This is a bad idea. These other things are worth a lot of marks, and students who do them last often run out of time and end up losing *many* marks. Many of these things can be written at the same time as the program, or before you have even begun to write the program. For example, you can plan a testing strategy and specific test cases before writing any code. In fact, this can save you time on the program itself: by thinking in advance of all the interesting test cases, you can avoid bugs.

The university's computer systems will be heavily loaded during the last few days preceding assignment due dates, and only a small fraction of the students enrolled in first-year courses can use the computing facilities at the same time. This is another reason for trying to complete assignments before the due date. **Overloaded facilities and one-day equipment outages will not be accepted as excuses for late assignments**. That includes printers that are very backed up or jammed immediately before the due time.

In a course as large as this one, it is inevitable that some assignments will be mislaid, and some marks will be recorded incorrectly. Keep *all* the runs of your programs until after you have received a final mark in the course; then when you say, "But I *did* do it, and I handed it in on time," you can prove it, and we can believe you — as we would like to be able to do. Further, if you have all the trial runs, no one can copy your program, and you can show us that you really did solve the problem yourself, in case there is a dispute over plagiarism.

It is also important to keep the graded copy of the assignment until after you have received a final mark in the course, in case of mistakes in recording. Marks will be posted throughout the term. It is your responsibility to confirm that your marks are correctly recorded.

11 Feedback on your assignments

The TAs work hard to get your assignments marked quickly, and to give you as much feedback as they can. However, their time is very limited, and you may not get as much feedback as you would like. If you would like to know more about what you did well and where you could have done better, consider bringing your marked assignment to an office hour of your professor.

Some parts of an assignment are simply right or wrong, but there are also parts that are marked more like an essay would be. Some students are surprised to learn this. For example, programming style and comments are not right or wrong; there are degrees of quality, and judging them is somewhat subjective. Don't expect to be told exactly where you lost every mark; a mark for something like style is usually assigned based on an *overall* impression of a program's style. Also, don't be shocked to get less than perfect, any more than you would be for getting less than perfect on an essay. A perfect mark for things like programming style or comments is rare, just as a perfect mark on an essay is rare.

12 Re-submitting assignments for remarking

It is inevitable that on each assignment a few students will disagree with their mark. This can happen for many reasons, including an error on the part of the marker, a misunderstanding, or different expectations about what a good solution should look like. In these cases, the student can ask that their assignment be remarked.

The best way to do this is to speak directly with the TA who marked your assignment. If you are unable to do so, or are unsatisfied with what the marker says, you may resubmit the assignment. To do this, fill out the "Request for Remarking" form on the course web site and give both it and the complete assignment to your professor. **You must do this within a week after the assignment is returned.**

13 How to avoid plagiarism

Assignments are given in order to help you learn, and the grades you are given are a measure of how much you have learned. For this reason, the assignments that you hand in must be your own work, and must not contain anyone else's ideas. Learning is like working out. You can only get in shape by working out yourself, and not by watching someone else work out. And the only way you can truly learn is by solving problems by yourself, not by watching someone else solve them for you.

Also, remember that we (the instructor and TA's) are here to help: if you feel lost or desperate, please come and talk to us instead of cheating!

What is plagiarism?

Plagiarism is a kind of fraud: passing off someone else's work or ideas as your own in order to get a higher mark. Plagiarism is a form of academic dishonesty and is treated very seriously. The assignments you hand in must be your own and must not contain anyone else's ideas.

Have a look at the Faculty of Arts and Science's Code of Behavior on Academic Matters, available in the Faculty of Arts and Science calendar on paper or at

`http://www.artsandscience.utoronto.ca/ofr/calendar/rules.htm#behaviour`

for a more detailed description of plagiarism and its consequences.

Guidelines for avoiding plagiarism

You may discuss assignments with friends and classmates, but only up to a point. You may discuss and compare general approaches and also how to get around particular difficulties, but *you should not leave such a discussion with any written material.* You should never look at another student's solution to an assignment on paper or on the computer screen, even in draft form.

To ensure that your solution is your own (and that you really do understand the material, which is the whole point), the actual coding of your programs, analysis of results, and writing of reports must be done individually. In particular, if you discuss the assignment with someone, don't take any notes, and make sure that you spend at least one hour doing something completely unrelated (like watching TV, reading a novel, playing a computer game, or shopping) before you attempt to write your solution. This way, your solution will truly be your own, which decreases significantly the chance that it will be similar to someone else's.

Remember that it is *not* difficult for the markers to detect undue collaboration. Also, for assignments submitted electronically, we will be comparing all submissions using special software that detects similarities between programs: simply changing variable names, format, or comments in your code will *not* fool this software, so you may as well write the code yourself!

Every year, a number of students commit plagiarism because they are desperate: the due date is coming up, they don't know how to do the assignment, and they figure that even if they get zero on the assignment because they cheated, it's no worse than getting zero because they handed nothing in. They're *wrong*: if you cheat, a notice will be put on your permanent record to indicate this, and further offenses in later courses will result in much more severe punishment. Also, the penalty for cheating can be worse than simply getting zero on the assignment. (For example, a further reduction of 10% on the final mark is common). More importantly, you really aren't learning material that you will need to know later on. If you cannot finish an assignment, at least be honest about it and hand in only your own incomplete work for part marks: you'll still get more marks that way than if you cheat.

Finally, note that it is also a serious offense to help someone commit plagiarism. *Do not lend your printouts, reports or diskettes, and do not let others copy or even read them*, even after the deadline for submission is passed (someone could hand in their assignment late). To protect yourself against people copying your work without your knowledge, retain all of your old printouts and draft notes until the assignments have been graded and returned to you, and *make sure that you change your password* the first time that you log on to your account. If you suspect that someone has stolen a printout or diskette, or has broken into your account, contact your instructor immediately.

Helping each other

Although you must not solve your assignments with the help of others, there are still many ways in which students can help each other and we encourage you to take advantage of this. For instance, you can go over difficult lecture or tutorial material, work through exercises, or help each other understand an assignment handout. This sort of course collaboration can be done in study groups or through the newsgroup.

14 Feedback for us on the course

Please feel free to get in touch with us with any suggestions or complaints that you have about any aspect of the course. Don't hesitate to let us know if there are aspects of the course that you particularly like, so that we can keep them that way, or if there are specific aspects that you dislike, so that we can make changes (or discuss with you our reasons for doing things that way).

It can be sometimes difficult to talk to someone directly about feedback you have for that person. So if you have suggestions or complaints about your TA, consider discussing it with your instructor. Your instructor can then contact your TA about it, while maintaining your anonymity. Similarly, if you have a specific suggestion or complaint about your instructor, consider mentioning it to your TA so that he or she can pass it on to us.

Note that this does not mean that we will accept unfounded complaints! If you have complaints or criticism that you are ready to discuss in a reasonable manner, that's great. If you are merely unhappy about something and have nothing constructive to say (e.g., "this course is terrible", with no thought about why or how it is terrible), then we suggest that you wait and think it over until you come up with something more concrete that we can work with.

But do think about it and let us know! Your feedback helps us make the course better.