CSc 451: The Icon Programming Language
Spring 2003

Instructor

Name:	William Mitchell
E-mail:	whm@cs.arizona.edu
Office Hours:	Tuesdays and Thursdays: Immediately after class until 4:30 Wednesdays: To be announced And by appointment
Campus office:	To be announced
Home office phone:	577-6431 (8 a.m.-10 p.m., seven days a week)
Home office FAX:	577-3159 (anytime)

Website

The website for the class is `www.cs.arizona.edu/classes/cs451`.

Textbooks

The following texts are required:

The Icon Programming Language, 3rd edition, Ralph E. Griswold and Madge T. Griswold. Peer-to-Peer Communications, 1996, ISBN 1-57398-001-3

Available as a PDF file at www.cs.arizona.edu/icon/books.htm and in bound form from Jeffery Systems at www.zianet.com/jeffery/books/.

Graphics Programming in Icon, Ralph E. Griswold, Clinton L. Jeffery, and Gregg M. Townsend. Peer-to-Peer Communications, 1998, ISBN 1-57398-009-9

Currently available from Jeffery Systems for only the cost of shipping.

Programming with Unicon, Clinton L. Jeffery et al., forthcoming.

Also available from Jeffery Systems in prepublication format and as a free PDF download.

Prerequisites

C Sc 342 required; C Sc 335 recommended; C Sc 372 not required

Course Description

This course is an in-depth study of the Icon programming language and the programming paradigms it supports. The full language is studied and emphasis is placed on non-conventional features such as the rich set of built-in types, goal directed evaluation, string scanning, co-expressions, and programmer-defined control operations.

In addition to the core language, Icon's graphical extensions, and Unicon, an object-oriented derivative will be studied. Elements from other languages such as Java, Python, Perl, and SNOBOL4 that provide alternative ways to support the same paradigms will be injected for contrast.

Potential additional topics include elements of the implementation of Icon, performance analysis, Jcon (a Java-based implementation of Icon), and other topics based on class interest.

By the end of the course students should be fluent in Icon and Unicon and be able to effectively apply the technology to a variety of problems. Students will also learn a number of programming techniques that can be applied with similar languages. Students with an interest in language design will have a broadened base of knowledge to draw upon.

Grading Structure

The grade for this course will be based on:

In-class examinations (2)	20%
Homework assignments	35%
Individual project	30%
Final examination	15%

The anticipated dates for the in-class examinations are Tuesday, February 25 and Tuesday, April 8. The final examination will be on Thursday, May 15, from 2:00-4:00.

Without prior arrangements, missing an exam will likely result in a grade of zero unless you are the victim of extraordinary circumstances, as determined by me.

Final grades will be based on a ten-point scale: 90-100 is an A, 80-89 is a B, and so forth. The lower bounds may be adjusted downwards to accommodate clustering of grades.

You are encouraged to contest any score assigned that you feel is not equitable.

Attendance is not recorded or enforced, but you are responsible for all material covered in lectures, appearing on handouts, or posted to the mailing list.

Assignments

There will be 350 points of homework assignments. Homework assignments will largely consist of programming problems but other types of problems may appear as well. I anticipate six major assignments and a handful of minor assignments but the final number may vary.

Each homework assignment will have a precise due date and time, typically on Thursdays at 6:00 p.m. As a rule, late homework assignments are not accepted and result in a grade of zero. Extensions may be granted to the class as a whole if problems arise with an assignment. Extensions for individuals may be granted in the case of extraordinary circumstances, as determined by me. You are encouraged to argue your case if you fall victim to circumstances beyond your control.

A "bug bounty" of one point of extra-credit will be awarded to the first person to report a given bug in an assignment. Bugs might take the form of errors in examples, ambiguous statements, incomplete specifications, etc. As a rule, simple misspellings and minor typographical errors won't qualify for extra credit, but each situation is unique. Any number of points may be acquired for a given assignment and will be added to the grade for that assignment.

My view is that it's a Bad Thing to give any credit for programs that don't work at all. Therefore, no solution, a solution that doesn't compile, and a solution that compiles but mostly doesn't work are equally worthless and will each earn a grade of zero. Examples of test data and expected output will usually be specified for each problem in an assignment. For some problems, reference versions of programs or procedures and/or a body of test data will be supplied on-line. Failure to get a solution working as shown in examples or as observable with a reference version and supplied data will typically result in a deduction of about one-third of the point value for that problem. For grading, solutions will typically be tested with all the supplied test cases and with additional test cases as well. If your solution works properly with all the supplied test cases you'll likely earn at least 85% of the point value, no matter what it does with the additional test cases.

My view is that programming assignments are to help you learn—I don't view an assignment as a take home exam. As a rule, you'll learn more if you can get through an assignment without asking for a lot of help. On the other hand, if you reach a point where you simply aren't making progress and you're out of ideas then that's the time to ask for help.

Individual Project

Thirty percent of the grade for the course will be based on a "large" programming project of each student's own design. Part of the project is identifying a problem to solve with Icon. The best candidates for projects are problems that have core functionality that is straightforward to attain and that can be built upon to reach a suitable level of complexity.

Some broad examples of potential projects are editors, browsers, games, visualization tools, web crawlers, text transformers/information extractors, mail/news readers, "movie" applications, and language interpreters. The use of graphics is encouraged but not required. If you make good use of Icon/Unicon and the available libraries, you can probably produce an interesting application with less than 2000 lines of code.

I will meet with each student to discuss project ideas and agree on sufficient functionality. It will not be possible to ensure that all projects are equally challenging; good planning and negotiation can result in considerable leverage.

Although the typical project will be an application of some sort, there is potential for a project to consist of design and implementation of a library.

Details on the project assignment will be forthcoming.

Computing Facilities

Access to the machine `lectura` will be provided for the development of homework solutions. If you are not familiar with the department's computing facilities please consult the Computing Information page: `www.cs.arizona.edu/indexes/computer.html`.

Rather than using `lectura`, some students may find it more convenient to develop solutions on a system at home or at a place of work. That's perfectly acceptable but if you do prepare solutions on a system other than `lectura` be sure to allow adequate time to transfer your solutions to `lectura` and test them there.

Office Hours, E-mail, and Telephone

I enjoy working with students whether it is in my office, on the phone, via e-mail, or walking across campus. My belief is that interaction with students outside the classroom is a vital aspect of teaching a course. I will do everything possible to make myself accessible to you.

I do most of my work at home and thus you'll find that I'm usually not in my office except during office hours, but if you do find me in my office, I'll usually have time to talk.

I prefer to conduct office hours in a group-style, round-robin manner. You needn't wait in the hallway if I'm working with another student; you may join us and listen in if you so desire. If several persons each have questions, I will handle one question at a time from each person in turn. I will often give priority to short questions (i.e., questions with short answers) and to persons having other commitments that constrain their waiting time. (Speak up when you fall in either of these categories.) If for some reason you would like to speak with me in private, let me know and I will clear the office.

You are encouraged to use electronic mail to communicate with me if you so desire. My

goal is to answer mail promptly. I give priority to well-focused questions. It's often the case that the task of developing a well-focused question will lead you to an answer on your own. Here are some examples of poorly focused questions: "Why doesn't `icont` work?" "Why does my program blow up?" "Why doesn't the `write` function produce any output?" If you use a windowing system, an appropriate clip of output can be very useful to me in diagnosing a problem. On a dumb terminal you can use the `script` command or open an Emacs shell window to capture the text of a session.

When on departmental UNIX machines you may address mail to me using simply `whm`. On other machines use `whm@cs.arizona.edu`. Mail from me may sometimes have a return address of `whm@mse.com`, but to receive the timeliest attention I recommend you use the address `whm@cs.arizona.edu`.

If you mail me with a problem and then solve it yourself before I reply, please send a follow-up note to let me know that the problem is resolved and that I needn't respond to your original note.

If you wish, you may contact me via my home office telephone: 577-6431. Feel free to try me anytime between 8 a.m. and 10 p.m., seven days a week. Some students will perhaps hesitate to call for fear they will be disturbing me but that concern is unfounded. If you have a question involving a specific piece of code, you should mail it to me and/or `turnin` a copy of the code before you call so that I may refer to it as we talk.

I am happy to call you if desired. If you leave a message for me with a telephone number, state the latest time when I can call you. If not specified, I'll feel free to call until 10 p.m. but not after.

Mailing List

A mailing list, rather than a news group, will be used as the electronic forum for this class. Messages from me might include information such as supplemental material, assignment corrections, lecture clarifications, due-date changes, etc. Please feel free to post messages such as requests for clarification on assignments, questions and comments about the lectures, interesting observations, etc. DON'T view the list as read-only—participate!

Note that when posting material related to assignments you should be very careful to not give away a solution or a portion thereof. If at all in doubt, send it to me alone.

The address of the class mailing list is `everybody-in-cs451@cs.arizona.edu`. An archive of messages will reside on `lectura` in `/home/cs451/mail`. Two ways to access the archive are "`pine -f /home/cs451/mail`" and "`/home/cs451/bin/showmail`".

Original Thoughts

In the movie comedy "Broadcast News" a 14 year-old high-school valedictorian receives a post-graduation ceremony beating from a group of bullies. After picking himself up, among the verbal spears he casts at them is, "You'll never have an original thought!" That notion of an "original thought" has stayed with me.

My hope is that you'll have some original thoughts as we move through the course material. I offer an award of a half-point on your final score for each original thought that you claim as such and that strikes me as significant. It might be an observation, an analogy, an interesting piece of code, or something else.

Academic Integrity

It is unfortunate that this section need be included but experience sadly shows that some students are willing to sacrifice their integrity to obtain a grade they have not earned. For those students who would never do such a thing, I apologize for the inclusion of this section.

Capsule summary: Don't cheat in my class and don't make it easy for anybody else to cheat.

You are responsible for understanding and complying with the University's Code of Academic Integrity. It can be found at w3.arizona.edu/~studpubs/policies/cacaint.htm. Among other provisions, the Code demands that the work you submit is your own and that submitted work will not subsequently be tampered with. Copying of another student's programs or data is prohibited when they are part of an assignment; it is immaterial whether the copying is by computer, Xerox or other means. Witting collaboration in allowing such copying is also a Code violation.

Violations of the Code will, at minimum, result in a negative grade being assigned. As a specific example, if it is determined that another student's work was copied verbatim for a 100 point assignment, the grade assigned for that assignment will be -100. An egregious first violation or any second violation will result in failure of the entire course along with possible further penalties at the discretion of the Dean.

In addition to ruining one's grade and damaging one's future, the processing of an academic integrity case can require hours of work by myself and others. I am happy to spend hours helping a student who is earnestly struggling with the material, but I truly loathe every minute I spend dealing with academic integrity cases.

It is difficult to describe in writing where reasonable collaboration stops and cheating begins, but here are some guidelines:

- I consider it to be reasonable to work together on assignments to get to the point of understanding the problem and the language facilities that are required for the solution.

- I consider it to be reasonable to help another student find a bug but only if your solution for that problem is further advanced than the one being examined. For example, if you haven't started on a solution, you shouldn't be hunting bugs in another person's version of that solution.

- It is surely a mistake to give another student a copy of a solution or a significant portion thereof.

- It is almost surely a mistake to submit a solution if you are unable to fully explain how it works.

- If your gut feeling is that you're cheating on an assignment or helping somebody else cheat, you probably are.

- If in the heat of the moment you submit for final grading a solution that is not yours and you later reconsider your action, you stand a better chance of leniency if you confess before your act is discovered. Conversely, further dishonesty when confronted, which in turn can increase the time expenditure, raises the likelihood of more extensive penalties.

Cheating almost always starts when one student has easy access to the solutions of another. You are expected to take whatever steps are necessary to guard your solutions from others in the class. For example, you should have a non-guessable password and never tell that password to another person. Hardcopy should not go into a recycling bin—take it home and dump it in a recycle bin when the course is over. If you and another student share a computing facility off-campus, perhaps as co-workers or roommates, using that system to develop solutions may be of very questionable merit, depending on the security facility on the system and the privileges of the other student. **Failure to take reasonable precautions to ensure the privacy of your solutions may be construed as witting collaboration.**