CSC 372: Comparative Programming Languages
The University of Arizona
Spring 2016

Instructor

William H. Mitchell (whm)
Office hours and contact info: *Posted on Piazza*

Schedule

Lectures: Tuesdays and Thursdays 14:00-15:15 in BIOW 208

Teaching Assistants

Undergraduate TAs:
Ben Gaska (bengaska)
Patrick Hickey (patrickhickey)

Office hours and contact info: *Posted on Piazza*

Website and handouts

We'll be using Piazza for announcements, discussions, and more. **If you haven't already signed up, do it now!** Go to piazza.com and follow their instructions.

All handouts will be posted on Piazza, on the Resources tab of the Resources page. All materials can also be directly accessed here: http://cs.arizona.edu/classes/cs372/spring16

Leftover handouts will be placed on the metal bookshelves near the freight elevator at the west end of the eighth floor of Gould-Simpson.

If you don't make use of paper handouts, let me know, and I'll reduce the copy count accordingly.

Prerequisites

CSC 127B or CSC 227. CSC major status.

Knowledge of Java, as covered in CSC 127A/B (or CSC 227), will be assumed.

The prerequisites are very modest but this is a 300-level computer science class with a lot of diverse content. As an analogy, imagine a 300-level math class with only high-school algebra as a prerequisite but that will venture into some exotic math that's not much like algebra at all. My task is to respect the prerequisites but also cover a body of material that's appropriate for a 300-level CS class.

Course Objectives

The purpose of this course is to explore some alternative ways of specifying computation and to help you understand and harness the forces that a programming language can exert. We'll spend a lot of time working with three languages: Haskell, Ruby, and Prolog. Functional programming will be studied using Haskell. Ruby will be used to explore imperative and object-oriented programming using a language with dynamic type checking. Prolog will transport us into the very different world of logic programming. You'll learn about some interesting elements of other languages.

Upon successfully completing the course you'll be around a "2" on a 1-5 scale (5 high) with Haskell, Ruby, and Prolog. You will understand the characteristics of the programming paradigms supported by those languages and be able to apply some of

the techniques in other languages. You'll have an increased ability to learn new languages. You'll have some idea about whether you want to pursue further study of programming languages.

Textbooks

No textbooks are required.

It is my intention that the lectures, handouts, and Piazza postings will provide all the information needed to successfully complete the course; but it's often good to see things explained in multiple ways. Below are some books that I recommend as good supplements but it's important to understand that the routes we'll be taking through the languages don't directly correspond to any book; the handouts you'll be getting are the primary "text".

Haskell:

I consider *Learn You a Haskell for Great Good!* by Miran Lipovača to be an excellent Haskell book and the best beginner's Haskell book on Safari.

I think of *Programming in Haskell*, by Hutton and on Safari, to be a good book once you've got the basics down, but I think it moves too fast to stand alone as a first book on Haskell, unless one has had prior exposure to functional programming.

Real World Haskell, by O'Sullivan, Stewart, and Goerzen, has some good introductory material as well as some interesting real-world examples. It's on Safari.

The Haskell School of Expression, by Paul Hudak, was one of the first Haskell books I owned. I didn't find it very helpful at the time but I've since grown to appreciate it. It's on Safari.

If you don't mind spending some money, then I recommend *Haskell: The Craft of Functional Programming*, 3rd edition, by Simon Thompson. It's very thorough; it doesn't make the sort of leaps that can leave beginners puzzled.

Ruby:

Safari has numerous Ruby books but it doesn't have the Ruby book I currently like the best, which is this:
Programming Ruby 1.9 & 2.0 (4th edition): The Pragmatic Programmers' Guide
http://pragprog.com/book/ruby4/programming-ruby-1-9-2-0

The Ruby book on Safari that I currently like the best is *The Ruby Programming Language* by David Flannagan and Yukihiro Matsumoto.

Prolog:

Safari has zero Prolog titles but an excellent text, *Programming in Prolog*, 5th edition, by Clocksin and Mellish, is available as a PDF that can be downloaded through the library. **Get a copy NOW**, in case licensing agreements change during the semester. Here's a link:
http://ezproxy.library.arizona.edu/login?url=http://dx.doi.org/10.1007/978-3-642-55481-0

Another Prolog book I really like is freely available on the net, albeit as pages scanned as images:
Prolog Programming in Depth, by Covington, Nute, and Vellino
http://www.covingtoninnovations.com/books/PPID.pdf

I'll make available on Piazza an OCR'd version of Covington's book with a hidden text layer added, so it can be searched.

Note that Safari can be accessed off-campus using the VPN or by using the library's proxy. Here's the proxy-based URL for Safari: http://proquest.safaribooksonline.com.ezproxy1.library.arizona.edu/ Hit it and then click "Start Using Safari" under "Academic License & Public Library Users".

Grading Structure

My goal is for **everyone** to earn an "A" in this course.

The final grade will comprise the following:

Assignments	60%
Pop quizzes	5%
Mid-term exam	13%
Final exam	22%

Final grades will be based on a ten-point scale: 90 or better is a guaranteed A, 80 or better is a B, and so forth. The lower bounds may be adjusted downwards to accommodate clustering of grades and/or other factors.

It is my goal that you will need to spend, on average, no more than ten hours per week, counting lectures, to learn the course material and get an "A" in this course. If you find that you're needing to spend more time than that, let's talk about it.

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

There is no attendance component in the grade—if you find that my lectures aren't worth your time, feel free to cut class!

The mid-term exam is tentatively scheduled for Thursday, March 10, the last class before Spring Break. It will cover all the Haskell material and some amount of Ruby.

My reading of http://www.registrar.arizona.edu/schedule2161/exams/tr.htm indicates that the final exam will be Monday, May 9, 2016, 3:30-5:30 pm, in our regular classroom. The final exam will be comprehensive but with more emphasis on Ruby and Prolog than Haskell.

My goal is to distribute language-specific points as evenly as possible between the three languages, across both assignments and exams.

Assignments

Assignments will largely consist of programming problems but other types of problems, such as short answer questions, essay questions, and diagrams, may appear as well. There will be a video project, too. As I write this I anticipate that there will be seven major assignments, one every two weeks, and some number of minor assignments but those counts and timings are subject to change. There will be a total of 600 points worth of assignments. Based on 600 total assignment points, a ten point homework problem corresponds to one point on your final average.

For programming problems great emphasis will be placed on the ability to deliver code whose output exactly matches the specification. Failure to achieve that will typically result in large point deductions, sometimes the full value of the problem.

My view is that it's a Bad Thing to give any credit for code that doesn't work. **Programs that don't compile or that mostly don't work will earn a grade of zero, regardless of how close to working they might be.** Additionally, non-general solutions, which might have the expected output "wired-in", will very likely earn a grade of zero.

Unless specifically requested in an assignment write-up, no comments are required in any code. Use comments as you see fit.

My view is that programming assignments are to help you learn the course material—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; but if you reach a point where you simply aren't making progress and you're running out of ideas or time, then you surely should ask for help.

Each assignment will specify a precise due date and time. **As a rule, late assignment submissions are not accepted and result in a grade of zero. There are no late days.**

Extensions may be granted to the class as a whole if problems arise with an assignment or with departmental or university

computing facilities.

Extensions for individuals may be granted in certain cases. **The key factor that leads to an extension is that due to circumstances beyond the student's control, the student's work is, was, or will be impeded, and it is impractical for the student to make up for the lost time.**

Accident, illness, friend/family crises, and significantly disruptive failures of technology are examples of circumstances that I generally consider to be beyond a student's control. On the other hand, for example, an extension due to assignments or exams in other classes is extremely unlikely. Travel, such as an interviewing trip, may merit an extension, but pre-trip discussion and approval of the extension is strongly recommended. Unexpected hours at a job, such as needing to fill in for a sick co-worker, may warrant an extension. **Ultimately, however, each situation is unique; you are strongly encouraged to contact me if you believe an extension may be warranted.** If you believe an extension is warranted, DO NOT work on an assignment (or even think about it) past the deadline; wait for an extension to be granted.

Extensions are granted in the form of an amount of time, such as eight hours. An eight-hour extension can be pictured as a count-down timer with an initial setting of eight hours. The timer runs whenever you are working on the assignment, whether that be typing in code or simply thinking about it.

Here's a scenario involving an extension:

> Your new laptop catches fire and burns itself into a cinder eight hours before midnight deadline on a Wednesday. I would likely grant you an eight-hour extension. On Friday, your next chance to get to the store, you get a new laptop and you're operational by Friday night. You might spend four hours Friday night working on the assignment, take off all day on Saturday and Sunday, spend two more hours on Monday and get absolutely stuck, and finish it up after a couple of questions during office hours on Wednesday.

You will be on your honor to keep written track of the time spent during an extension and not exceed the amount granted.

All holidays or special events observed by organized religions will be honored for those students who show affiliation with that particular religion. Absences pre-approved by the UA Dean of Students (or Dean's designee) will be honored.

Bug Bounties

A "bug bounty" of one assignment point of extra credit will be awarded to the first student to report a particular 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 a bug bounty point; but each situation is unique—you are encouraged to report any bugs you find. Any number of bug bounty points may be earned for an assignment and will be added to the grade for that assignment.

Bug bounty points may also awarded for bugs in the slides, my Piazza postings, quizzes, exams, and this syllabus. Such points are added to the next assignment.

Please report bugs by mail or IM, not via Piazza, to give me a chance to filter any false positives.

Don't interrupt lectures to point out minor bugs on slides but do speak up if an error seems serious. My usual practice is to only put code and commands onto a slide with copy/paste after testing it first, but if you see an error in code or commands, speak up.

Quizzes

There will be some number of "pop" quizzes. Quizzes will typically be a handful of questions, be allocated three minutes or less, and be worth 1-5 quiz points. There will be a total of 50 quiz points, corresponding to 5% of the final grade. Quizzes may be conducted at any time during the class period. In some cases a quiz may be given simply to see what portion of the class grasped some just-presented material and be graded on the basis of participation rather than correctness.

Computing Facilities

You're free to develop solutions for programming problems on any machine you wish but your solutions will be graded on the

CS machine named "lectura", so it's important to test your solutions on lectura before you submit them for grading.

By virtue of being enrolled in this class you should already have a CS computing account with the same name as your UA NetID but a password that's likely different (a good practice). With that account you can login on any of the CS instructional machines, including lectura. The files in your home directory tree are stored on a server and appear the same no matter which CS machine you're logged into.

My UNIX slides from 352 in Fall 2015 have details about logging into lectura, resetting your password, and more. Those slides are here: http://www.cs.arizona.edu/classes/cs352/fall15/unix.pdf. See slides 14-21 for details about logging in. Slides 74-102 discuss options for editing files on lectura and tools like WinSCP and DropSync, to keep directories on lectura synchronized with corresponding directories your laptop.

The CS computing facilities are described in http://cs.arizona.edu/computing/facilities. The FAQs at http://faq.cs.arizona.edu have lots of answers.

Office Hours

I truly enjoy working with students. I believe 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 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; I will clear the office. Or, make an appointment to meet with me outside of office hours.

Students who make proactive, not reactive, use of office hours usually achieve the best results. Proactive use of office hours includes asking questions about material on the slides and in the texts, asking questions about how to better use tools, discussing how to approach problems, etc. If you're familiar with Covey's *The Seven Habits of Highly Effective People* you might recognize those as "Quadrant II" activities—important, but not urgent.

Reactive use of office hours is typically centered around simply getting code to work one way or another (Covey's "Quadrant I"—important and urgent).

Piazza

As mentioned above, we'll be using Piazza. All students are strongly encouraged to participate freely. I hope you'll post questions and comments related to the course material, recommendations for handy tools, URLs for interesting web posts and videos, and whatever else you think is worth mentioning as long as it relates to the course material in some way. Posts about CS-related job opportunities, and events like programming contests and hackathons are welcome, too.

The initial post in any discussion thread that I start is considered to be part of the course material—you are responsible for reading each and every one, and taking action when appropriate. For example, I might ask you to read an article, or experiment with a web site of interest. You may learn more by reading follow-ups by classmates and by me, but at exam- or quiz-time I won't expect you to have read each and every follow-up.

When answering questions on Piazza, the TAs and I will give priority to well-focused questions. And, it's often the case that the task of developing a well-focused question will lead you to an answer on your own.

You can make anonymous posts, but be aware that such posts are only anonymous to classmates, not me or the TAs.

Needless to say, BE CAREFUL that Piazza posts don't give away the solution for a problem. If at all in doubt, send mail instead. (See **Mail** below.) A common case for a post that gives away a solution is when a student is very close to the solution, but doesn't realize it, perhaps because a tiny piece is missing. Haskell and Prolog solutions are often very short, sometimes just a line or two, so posting any code or describing any elements of a solution is very risky.

The typical penalty for giving away a problem is that the guilty student will be required to create an equally great problem as a replacement.

Piazza's "Private posts"—posts seen only by me and the TAs—are disabled because every once in a while somebody forgets to mark an intended private post as private, and everybody sees their code, or more.

Part of the idea of Piazza is to facilitate students helping each other, so don't hesitate to answer questions when you're so inclined. If a post is plain wrong, I'll add a correction. If a post is great, I'll endorse it. If I'm silent, then its quality is probably somewhere in the middle.

If you post a question and later solve it yourself, save everybody some time by posting a follow-up saying the question has been resolved.

Mail

For private communication with the TAs and me, mail to `372s16@cs.arizona.edu`. To ensure quality and accuracy, I want to see all mail between students and TA's. If you're replying to a TA's response to a question, use your mailer's "Reply All" to follow the Cc:'s.

If an email message asks a question or raises a point that I think should be shared with the class, I'll share it. If I think the post deserves kudos, I'll identify the author unless the post specifically requests anonymity, in which case I'll say something like "A student wrote..."

Instant Messaging

IM can be very effective, especially for short questions. Skype is my IM tool of choice. My id on Skype and other popular IM services is `x77686d`. If an IM session starts to run long, I'll often suggest adding in voice and maybe http://join.me, too. Don't pay much attention to my Skype status—sometimes I'm not available when I'm "Online", and at other times I might be invisible but happy to answer. I recommend you start with an "ayt" ("Are you there?") and then follow with your question if I respond.

It's hard to characterize what's best posted on Piazza versus asking on IM so I won't wrestle with that question here.

Original Thoughts

In the movie comedy "Broadcast News" a 14 year-old high-school valedictorian receives a post-commencement beating from a group of bullies. After picking himself up, one of the things he shouts to wound his departing attackers is, "You'll never have an original thought!" That notion of an "original thought" has stayed with me. I hope that you'll have some original thoughts during this semester.

I offer an award of a half-point on your final average for each Original Thought. Observations, analogies, quotable quotes, and clever uses of tools and language constructs are some examples of things that have qualified as Original Thoughts in my classes. Note that an Original Thought does not need to be something that's probably never been thought of before; it just needs to be something that I consider to be reasonably original for you.

Sometimes I'll point out that something you said in class or office hours, or wrote in your observations for an assignment, strikes me as an Original Thought. If you self-identify a potential Original Thought, let me know. In some cases I'll see a glimmer of an Original Thought in something, and encourage you to explore the idea a bit more.

Of course, an Original Thought needs to be something you've thought up yourself—don't send in something you found elsewhere, like a quote, just because it strikes you as being original!

The "bar" rises with each Original Thought for an individual—it's harder to earn a second Original Thought than a first, and a third is harder still.

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 cheat, I apologize for the inclusion of this section.

Capsule summary: Don't cheat in my class. Don't make it possible for anybody else to cheat. One strike and you're out!

You are responsible for understanding and complying with the University's Code of Academic Integrity. It can be found at http://deanofstudents.arizona.edu/codeofacademicintegrity. 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, photograph or other means. Allowing such copying, thus facilitating academic dishonesty, is also a Code violation.

In addition to ruining one's grade and damaging one's future, the processing of an academic integrity case requires hours of work by myself and others. I am happy to spend hours helping a student who is earnestly trying to learn the material, but I truly loathe every minute spent on academic integrity cases. Even the very simplest of cases often takes 3-4 hours of my time, not to mention the time of office staff and others.

A violation of the Code of Academic Integrity will typically result in ALL of the following sanctions:

1. **Assignment of failing grade for the course**
2. **A permanent transcript annotation, like "FAILING GRADE ASSIGNED DUE TO CHEATING"**
3. **Recommendation of a one-semester suspension from the university**

When a student is caught cheating and I remind them of the above sanctions they often respond by sending me an email message that is hundreds of words long. They usually start by saying that a failing grade is surely sufficient punishment and that they have learned their lesson; they won't cheat again. They go on to say that some employers won't consider hiring a student whose transcript shows evidence of cheating. (True.) Then they talk about how a suspension from the university will cause them to lose scholarships, job offers, visas, and more. Some mention ill or aging family members who might not live to see them graduate if a suspension is imposed.

If you ever find yourself considering whether cheating is worth the risk, first imagine how the three sanctions above would impact you and your family. I'm willing to go to great lengths to help students learn the course material but if a student cheats, I'm equally willing to impose great penalties.

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

- It is surely cheating to submit code or text that was not written by you. Exception: If I work you through any part of a solution, either individually or in a group setting, you may freely use any code developed in that process. However, you may not pass along that code to anyone else.

- It is surely cheating to send another student any portion of a solution.

- I consider it to be reasonable to work together on assignments to get to the point of understanding the problems and the language or library elements that are required for solutions.

- I consider it to be reasonable to help another student find a bug if (1) you've finished that problem, and (2) your help consists of asking questions that help the other student to see the problem for themselves. If you find yourself about to dictate code to a classmate, STOP! (Note that occasionally during office hours you will see me dictate code to a student but that's because I've decided that's the best way to help them learn given the full situation at hand. You do not have that prerogative.)

- I consider it to be reasonable to exchange test cases unless test cases are one of the deliverables for a problem.

- If you receive help on a solution but are unable to fully explain how it works, it is surely a mistake to submit it as your

own work.

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

- **The vast resources of the Internet raise some interesting issues. See the section below on *Google, Stack Overflow, and more*.**

- If in the heat of the moment you submit a solution that is not fully yours, or give your work away, and you later reconsider your actions, you and any recipients will at worst lose points for the work involved if you confess before your act is discovered. Conversely, further dishonesty when confronted, which invariably increases the time expenditure, raises the likelihood of more extensive penalties, like a recommendation for a multiple-semester suspension or outright expulsion from the University.

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 an unguessable, uncommon password and never share it. Hardcopy should not go into a recycling bin—take it home and dump it in a recycle bin when the course is over. Personal machines, be them laptops or home desktops, should be behind a hardware firewall or running a software firewall.

Failure to take reasonable precautions to ensure the privacy of your solutions may be construed to be facilitating dishonesty, a Code violation. For example, having a weak password such as, but not limited to, your first name, your last name, your phone number, your initials, your sweetie's name, your pet's name, a reversed name, a sequence of consecutive keys, or a common password could be viewed as facilitation of dishonesty. For one list of common passwords see http://cs.arizona.edu/~whm/common-passwords.txt (If you know of a bigger, better list, let me know.) Hint: Don't look for your password by doing `grep MY-PASSWORD common-passwords.txt`—commands like `ps` and `w` show the arguments for currently-running commands!

For some interesting reading about password choice, see http://wpengine.com/unmasked.

Leaving a logged-in and unlocked machine unattended in the presence of another person, even a friend, is questionable. Use a screen locker!

Be very careful when typing a password in the presence of others, be them friends or strangers. The single worst cheating case I've ever handled started when the password of a two-fingered typist was surreptitiously observed. **Don't hesitate to ask someone, even a friend, to turn their head when you're typing your password.**

Malware is a Federal Crime

"Malware, short for malicious software, is any software used to disrupt computer operation, gather sensitive information, or gain access to private computer systems."—Wikipedia

In various situations my TAs and I will be running your code. Do not be tempted to slip some malware into your code, even for "harmless fun". Introducing malware into a computer system is a federal crime—see http://fas.org/sgp/crs/misc/97-1025.pdf. I will request permanent expulsion from the university for any student who submits malware, and recommend that the university notify the FBI for further action.

Google, Stack Overflow, and more

Things like Google and Stack Overflow are incredible resources for working professionals but when used to directly search for solutions for problems on an academic assignment they can cause learning and/or creative opportunities to be forever lost.

My expectation is that the material presented in class, practice via exercises, suggested readings, resources cited, and any prior assignments provide everything you need to do every problem on each assignment. I challenge you to limit your Googling to searches that expand your knowledge of the material, and not try to dig up quick solutions to problems. (And, of course, using any code found in such searches would be cheating.)

Posts on websites, IRC channels, mailing lists, etc. that solicit the answer for a problem or a significant piece thereof

will be considered to be cheating! Example:

> "*I'm learning Haskell and trying to write a function that returns True iff the parentheses in a string are properly matched. Any suggestions?*"

Accessibility and Accommodations

It is the University's goal that learning experiences be as accessible as possible. If you anticipate or experience physical or academic barriers based on disability, please let me know immediately so that we can discuss options. You are also welcome to contact Disability Resources (520-621-3268) to establish reasonable accommodations.

Please be aware that the accessible tables and chairs in the classroom should remain available for students who find that standard classroom seating is not usable.