CPSC 312 FUNCTIONAL AND LOGIC PROGRAMMING
WINTER TERM I 2015 (3 Credits)
SYLLABUS

Time and Place

Tuesdays & Thursdays
2 to 3:30pm
Hugh Dempster 310

Instructor

Sara Sagaii
sarams@cs.ubc.ca
Office: ICCS 187
Phone: 604-827-5554

Office Hours

Tuesdays 10-11am
Thursdays 9:30-10:30am

Teaching Assistants

Rui Ge <rge@cs.ubc.ca>
Susanne Bradley <smbrad@cs.ubc.ca>
Khurram Ali <khurram.ali@alumni.ubc.ca>

Course Webpage (Piazza): piazza.com/ubc.ca/winterterm12015/cpsc312/home
Note on anonymity: Please be aware the Piazza is an American service hosted in the United States. Registration on Piazza means that you consent to allowing Piazza to store your first name, last name, and email address. Some of you may choose to preserve your anonymity so that none of your personal information is stored in the United States (If you already have a gmail, hotmail or other US hosted email services, this is essentially moot). If you'd like to preserve your anonymity on Piazza, please contact cpsc312-admin@cs.ubc.ca and they will let you know what to do.

(Official) Course Overview

Principles of symbolic computing, using languages based upon first-order logic and the lambda calculus. Algorithms for implementing such languages. Applications to artificial intelligence and knowledge representation.

Unofficial (Kurt Eiselt's) Description

If you're a UBC student who has fulfilled the prerequisites for this course, then you've done some programming in Java or C++. These programming languages, like many languages, are imperative programming languages. Imperative programs are written as sequences of instructions that change or mutate program state. The instructions themselves correspond fairly closely to machine-level instructions that move data from memory to registers, modify the data, and then

move the modified data back to memory. An imperative programming language thus constrains the programmer to define a solution to a problem in terms of the computer's low-level architecture. Wait, you say, aren't Java and C++ object-oriented languages? Well, sure, adding concepts like classes and objects to a language puts some useful distance between the programmer and the computer architecture and provides additional flexibility when considering possible solutions, but when you get down to the task of writing Java methods, you're writing imperative programs. Imperative programming languages are good for tackling many computational problems but they are by no means the best tools for solving all computational problems.

CPSC 312 exposes students to two other programming paradigms, the functional programming paradigm and the logic programming paradigm, each of which provides a very different perspective on computation than the imperative paradigm. In CPSC 312, the functional languages are represented by Haskell and the logic languages are represented by Prolog. Each of these languages will require you to think about computation in ways you probably never have before. That's really the point of this course. As the noted computer scientist Alan Perlis once said, "A language that doesn't affect the way you think about programming is not worth knowing."

As the course title says, this is a "programming" course, so if you're not fond of programming, you might look elsewhere for a course to fill out your schedule. Learning two (or maybe three, or even four, depending on how things go) new programming languages while trying to integrate two very different styles of thinking with what you already know will, for some students, require far more practice than is provided by the homework assignments. If you don't like programming, then you're less likely to practice as much as you need to, which may in turn have an unfortunate impact on your exam performance. I'm not trying to scare you away. I just want you to understand that this course can be challenging.

Learning Objectives

1. To learn about non-imperative programming paradigms, their merits as well as their shortcomings. To provide the students with a clearer, deeper, broader, and more critical understanding of programming languages.
2. Students should be able to compare and contrast the imperative paradigm with functional and logic programming paradigms.
3. Learn to think in the functional and logic paradigms and write Haskell and Prolog programs of basic to intermediate levels of difficulty.
4. Provide the students with a historical, theoretical and industrial context to understand the purpose and application of these paradigms within the discipline of computer science as well as the software industry.

Textbooks:

The Art of Prolog (Second Edition), Leon Sterling and Ehud Shapiro, MIT Press, 1994.

Haskell: The Craft of Functional Programming **(Third Edition)**, Simon Thompson, Addison-Wesley, 2011.

Grading

Assignments	15%
Projects	30%
Midterm exam	20%
Final exam	35%

Note: In order to pass this course you must obtain a 50% overall mark, and you must pass the final exam as well as each of the two projects.
Participation: 5-10% bonus marks will be rewarded to active participation in the class and/or high quality contribution to piazza discussions (Your instructor reserves the right to reward this bonus based on her judgement and consultation with the TAs).
Disputing Grades: If you need to dispute a project or assignment grade, please contact your TA first. If the dispute cannot be resolved, your TA will contact the instructor. Midterm grades can be disputed up to a week after the grade is received.

Academic Honesty

Please take the time now to familiarize yourself with the University policy on academic misconduct. You are responsible for understanding and complying with this policy:
UBC policy on Academic Misconduct
(URL if the link is not working: http://www.calendar.ubc.ca/vancouver/index.cfm?tree=3,54,111,959)

Collaboration and Discussion Policy

Team collaboration policy will be explained in length when the projects are released. Assignments are to be done individually. If your solutions are too similar to that of another student, it will be detected automatically and you will both lose marks. Questions about any of the assignments or projects can be posted to Piazza or emailed to TAs. Please DO NOT POST SOLUTIONS TO PIAZZA. TAs and the instructor will be regularly monitoring the discussion board and posting solutions or parts of your code in such a way that reveals a solution will cause you to lose mark. When in doubt send it privately to instructors.

Late Submissions, etc.

Generally speaking, late assignments will not be accepted or marked. However, if you know beforehand that due to circumstances you may not meet a certain deadline, talk to your instructor at least 5 days prior to the deadline and explain your situation. An extension of 1 or 2 days is not guaranteed but may be considered based on your individual situation. If you are having trouble keeping up, the best thing to do is to come to an office hour and talk to a TA or the instructor. Alternatively, you can speak with an academic advisor or if need be a counsellor (http://students.ubc.ca/livewell/services/counselling-services). The worst thing is to simply let a deadline pass.

Course Schedule

This schedule is tentative and subject to change.
Week 1 (Sep 10)
Introduction to Declarative Programming

Week 2 (Sep 15, 17):

(Beginning PROLOG)

Ch. 1: Basic Constructs

Ch. 2: Database Programming

Ch. 3: Recursive Programming

Ch. 4: The Computational Model of Logic Programs

Week 3 (Sep 22, 24)

Ch. 6: Pure Prolog

Ch. 7: Programming in Pure Prolog

Ch. 8: Arithmetic

Week 4 (Sep 29, Oct 1)

(parts of:)

Ch. 11: Cuts and Negation

Ch. 12: Extra-Logical Predicates

Ch. 13: Program Development

Week 5 (Oct 6, 8)

(parts of:)

Ch. 14: Nondeterministic Programming

Ch. 15: Incomplete Data Structures

Ch. 16: Second-Order Programming

Ch. 19: Definite Clause Grammars and Natural Language

Week 6 (Oct 13, 15)

Midterm exam

Week 7 (Oct 20, 22)

(*Beginning Haskell)*

Ch. 1: Introducing Functional Programming

Ch. 2: Getting Started with Haskell and GHCi

Week 8 (Oct 27, 29)

Ch. 3: Basic Types and Definitions

Ch. 4: Designing and Writing Programs

Week 9 (Nov 3, 5)

Ch. 5: Data Types, Tuples and Lists

Ch. 6: Programming with Lists

Ch. 7: Defining Functions over Lists

Week 10 (Nov 10, 12)

Ch. 10: Generalization: Patterns of Computation

Ch. 11: Higher-Order Functions

Week 11 (Nov 17, 19)

Ch. 17: Lazy Programming

Ch. 18: Programming with Monads

Week 12 & 13 (Nov 24, 26, Dec 1, 3): TBD.