CSCI 1103 Computer Science 1 Honors
CSIC 2103 Functional Programming

Fall 2020

Instructor Muller

Syllabus

Welcome to CSCI 1103/CSCI 2103. This course serves dual purposes both as the Honors introductory course for Computer Science and as a 2000-level elective in functional programming. The 2000-level version of the course is a good option for Bachelor of Arts in CS majors looking to fulfill the 2000-level elective requirement. Of course this is one course, but students enrolled in the 2000-level version will have additional work requirements on several problem sets.

This course is an introduction to the art and science of computer programming and to some of the fundamental concepts of computer science. Students will write programs in the OCaml dialect of the ML programming language. ML is a modern programming language featuring polymorphic static typing, automatic memory management and a robust module system. ML is *value-oriented* — computation is driven by an imperative to find the *value* of an expression. ML is often also called a *functional* programming language because of its emphasis on functions.

Good program design methodology will be stressed throughout the course. There will also be a study of some of the basic notions of computer science, including computer systems organization, files, and some algorithms of fundamental importance.

Course Goals

The main goal of this course is to help the student develop an understanding of *computation* and to help them master the art of designing algorithms, and developing the programs that implement them. Important parts of the latter include documenting and testing the program.

Students will learn how to decompose problems into specific subproblems, write an algorithm to solve a specific problem, and then translate that algorithm into an OCaml program.

Basic Information

The homepage for CSCI 1103 is on GitHub https://github.com/BC-CSCI1103/f20. (Students should certainly bookmark this link.) CSCI 1103 meets on-line via

zoom for 50-minutes on Mondays, Wednesdays and Fridays at 11AM. CSCI 1103/2103 also has a 50-minute lab.

Attendance at both the class meetings and the labs is critical, as all new material will be presented there.

Course Web Site

Please bookmark the course homepage which hosted in GitHub:

`https://github.com/BC-CSCI1103/f20/`

We'll use this site heavily throughout the semester and most of the course materials will be distributed through this site. Problem sets are to be submitted through this site. Note that we will not have a course homepage on Canvas.

Staff

Instructor: Robert Muller

email:robert.muller2@gmail.com

office: Zoom

hours: Tuesdays 10AM - 12PM, Wednesdays 1PM - 3PM and by appointment on Tuesdays through Fridays, as available.

phone: 617-552-3964

Head Teaching Assistant: Gavin Bloom

email: bloomga AT bc DOT edu

office: Zoom

hours: Thursdays 1PM - 4PM.

Teaching Assistant: Emma Sabbadini

email: sabbadie AT bc DOT edu

office: Zoom

hours: Tuesdays 7PM - 8PM, Fridays 12PM - 2PM.

Teaching Assistant: Callie Sardini

email: sardinac AT bc DOT edu

office: Zoom

hours: Wednesdays 6PM - 8PM, Sundays 4PM - 5PM.

Lab Times

The labs are run by the course Teaching Assistants. They provide a great opportunity for you to work with a partner and practice with new material on some fun problems in a supervised setting. Regular attendance at labs is strongly encouraged.

1. CSCI100802 (Gavin Bloom) Tuesdays 5PM - 6PM,
2. CSCI100803 (Emma Sabbadini) Tuesdays 6PM - 7PM,
3. CSCI100804 (Callie Sardina) Wednesdays 5PM - 6PM.

Problem Sets

Each week you will be assigned a problem set. Some problem sets are *open*, they can be done in pairs, and some are *closed* they must be done individually. Unless otherwise specified, all problem sets are due on Mondays at midnight. The single best indicator of success for computer science is *starting problem sets early*.

Problem sets should be submitted for grading by using a `git push` command to upload them to the course GitHub site. Problem sets cannot be submitted as email attachments. Attempts to submit problem sets as email attachments will not receive an email reply indicating that the attempted submission failed.

Topics

Roughly construed and subject to variation.

1. Overview, administration, OCaml setup and introduction. Types, literals, operators and expressions. Simplification, and values. Functions, libraries, function calls, function definitions.
2. More on function definitions and calls, variables and the simplification/substitution model of computation. let-bound variables, binding patterns, type notation. Using the graphics system. Pattern matching, match expressions. Branching, repetition.
3. More on repetition, primality, tessellation, A bisection algorithm for square roots.
4. Lists and tuples, appending, reversal, work/complexity. More on lists, linear search, association lists. Insert, split, partition and merge.
5. Sorting: insertion sort, mergesort, quicksort.
6. More work with lists and tuples, making change, positional addition. First Exam.
7. Functions are values, mapping and folding/reducing.
8. Type definitions, record types, sum types, option types. Binary search, binary search trees.
9. Numeral systems, binary, logic gates, a full-adder, a ripple-carry adder.
10. Machines: storage/memory, bytes and hexadecimal numerals. A storage model of evaluation. Storage diagrams. Machine architecture, SVM, assembly code.
11. Imperative coding, mutation and block-storage (arrays), imperative repetition idioms while and for. Working with 1 and 2D arrays.
12. Backtracking, 8-queens. Second Exam.
13. Strings, text, file IO and command-line arguments. Applications in bioinformatics, Markov models.
14. Defining new types in OCaml and Java.
15. Defining new types in OCaml and Java. Review and wrapup.

Exams

There will be three *closed* problem sets that serve as take-home exams.

Reading

There is one textbook for the course *OCaml from the Very Beginning* by John Whittington. There are two other books listed on the course homepage. We will use extensive code and lecture notes which will be posted to the course web site.

Grading

Your grade for this class will be computed on a 200 point scale from a combination of your open problem sets, your closed problem sets, and participation work. Participation is largely based on effort (not correctness). Lab work and Piazza involvement will be incorporated into the participation score. Final grades are computed, as follows:

- Seven open problem sets, these will account for 110 points;
- Three closed problem sets, 60 points;
- Class, lab and piazza forum participation, together, these account for the remaining 30 points.

Late Homework Policy

Homework is due on the day indicated at midnight. *This is a strict deadline.* Homework submitted at 12:01AM is one day late as is homework submitted 23:59 late. Late homework is penalized 25% per 24-hour period.

In the case of medical exigencies, students may petition the the Instructor for an extension. Medical problems or family emergencies are the only conditions under which extensions will be granted.

Honor Code

All solutions and code should be produced by you alone, or by you and a partner, where appropriate. For pair-programmed assignments, each partner needs to submit the assignment and each needs to acknowledge the other partner when submitting.

You may discuss algorithms at a high level with any student in the class. You may also help any student find a small bug in their code. However, you may not copy solutions from anyone, nor should you collaborate beyond high-level discussions with anyone who is not your partner. For pair programming problems, you must follow the guidelines given above.

If you have any questions about what behavior is acceptable, it is your responsibility to come see one of the instructors before you engage in this behavior. We are more than happy to answer any questions you may have.