



- Given an imperative program and a set of eager/lazy parameter-passing mechanisms, the student will be able to simulate, for each mechanism, the sequence of updates that take place in memory as the program executes.
- Given the description of an operation applicable to an infinite data structure, the student will be able to program this operation in ML using lazy evaluation.
- Given a functional language with higher-order functions, the student will be able to simulate recursion using the Y combinator.
- Given an interpreter for a language with pointers, the student will be able to add recursion to the interpreted language via the technique called "tying the knot."
- Given a working interpreter for a programming language, the student will be able to adapt the interpreter to a similar language with a different concrete syntax.
- Given a working interpreter for an imperative programming language, the student will be able to implement an interpreter for an enhanced language with additional features (such as a new data type or a new language construct), or different semantics (such as a different parameter-passing mechanism).
- Given a Java program that exhibits polymorphism, the student will be able to describe the behavior of the program using dynamic dispatch.
- Given a working interpreter for a simple object-oriented programming language, the student will be able to extend the interpreter/language with additional features such as new operators (e.g., "instanceof"), new access modifiers (e.g., private), or new object-oriented language features (e.g., method overloading).
- Given a problem description, the student will be able to program a parallel algorithm to solve the problem more efficiently than a sequential program would.

**Topic Coverage:** We will cover the following topics:

- Formal syntax (BNF, EBNF)
- Introduction to the ML programming language
- Type systems and type inference
- Scope and higher-order functions
- $\lambda$ -calculus (if time permits)
- Interpreters
- Evaluation order and parameter passing
- Object-orientation
- Parallel programming

### Course Grading Policy

Your final grade for this course will be based on three components, namely exams, assignments, and quizzes. Within each component (for example, the *exams* component), all items (for example, all exams) will have equal weight. In other words, each component grade will be computed as the average of all items in the component (however, see important note at the top of the next page). Your overall numerical grade for the course will be computed as the weighted sum of the component grades using the following weights:

Component	Weight
Exams (3)	45%
Assignments	45%
Quizzes	10%

**Important note:** If you receive a zero on 3 or more quizzes, your **overall quiz grade** will automatically be reduced to 0.

Finally, your letter grade for the course will be computed as follows:

Numerical Score	Grade	Numerical Score	Grade
92 and above	A	Between 72 and 79	C
Between 89 and 92	AB	Between 69 and 72	CD
Between 82 and 89	B	Between 60 and 69	D
Between 79 and 82	BC	Below 60	F

While this overall grading scheme is fixed, I will be happy to discuss any issue you may have with individual grades. If you notice a mistake or have a question regarding a specific grade, please come and talk to me *as soon as possible*. Do not wait until the end of the semester to bring up grading issues. Also, I will *not* be available to discuss grades after the end of the final week.

### Attendance and Participation

You are expected to not only attend **every** class meeting but also to come **prepared** for and **participate** actively in it. Necessary preparation requires you to have studied and assimilated the material covered in previous sessions, to have met with me outside of class to discuss any questions you may have, and to have completed the assignments on time. **It is hard to imagine how a student could do well in this course while missing classes or attending them unprepared.** On the positive side, I have high expectations for my students and will always support and encourage you. I **strongly encourage you to ask any question** or raise any issue you have with the course either during or at the end of class, or during my office hours. I will also gladly meet with you by appointment. Send me email to make an appointment. While I will meet with you as soon as my schedule permits, do not expect me to be widely available just before an assignment is due.

### Late Submissions

I will describe the submission procedure for your assignments when the time comes. However, let me point out right away that each assignment will come with a deadline (day and time) after which any submission will be considered late. The late-submission policy works as follows:

Turned in	Penalty
On due date but after deadline	10%
One day late	25%
Two days late	50%
Three days late	75%
Four days late	100%

Note that assignments more than three days late receive no points. Weekend days and holidays count as "regular days" when computing late penalties. Extensions on assignments may be granted at the discretion of the instructor if you provide a valid justification (in the form of a written excuse from a medical doctor or the Dean of Students Office) **before** the due date. Late

submissions can easily be avoided by starting to work on the assignment right away and asking for help early if you get stuck.

If you miss a scheduled exam, you **may** be able to take a make-up exam provided you give the instructor a valid justification (see above), ahead of time if possible. Only one make-up exam will be given. It will be a comprehensive exam scheduled at the end of the semester. Similarly, there will be no make-up quizzes unless the instructor is provided with a valid justification (see above) for your absence on the day of the quiz.

### **Bonus Points**

Since late submissions are penalized, it is only fair to reward early ones as well. Therefore, if you submit a **CORRECT** assignment at least one full day early, you will earn three bonus points for every full 24-hour interval (only up to 3 such intervals will be counted, for a maximum of 12 bonus points) between your submission's time stamp and the deadline time. Furthermore, there will be other opportunities for bonus points throughout the semester.

### **Collaborating versus Cheating**

Unless otherwise stated in the assignment, all submissions must be the work of a single student (the one whose name appears on the submission, that is). While it is acceptable to discuss the assignments at a high level (for example, at the design level) with others, you must submit your own work. You may not “borrow” any piece of code or design of any length from someone else, unless you can live with a zero and the other potential academic sanctions of cheating (see the [UWO Student Discipline Code](#), Chapter UWS 14).

In conclusion, remember that computer science classes require a lot of work in addition to active participation in class. It takes considerable practice to develop the technical and analytical skills targeted by this course. You will need to spend **at least (and typically more than) three hours of effort outside of class for each in-class hour**. Having said this, I expect every hardworking student to do well in this course.

**Have fun this semester and good luck!**