Programming Languages
(Comp Sci 331, Section 001)

Instructor: Erik Krohn
E-mail: krohne@uwosh.edu
Text Message Only: 920-644-3745
Class Information: Monday, Wednesday, Friday: Halsey 57, 3:00pm – 4:00pm
Office Location: 216 Halsey
Office Hours: Monday, Wednesday, Friday: 2:00pm – 3:00pm
Tuesday: 2:00pm – 4:00pm
Prerequisites: A grade of C or better in CS251 and CS271
Course Website: http://www.uwosh.edu/d2l/

Course Information
A study of programming languages. Topics covered include: formal syntactic description, methods of implementation, and language features such as recursion, block structure, string processing, and list processing. Specific high level programming languages are studied to demonstrate the use of these language features.

Course Website
You should check d2l on a regular basis - it will contain lecture notes, handouts, assignments, announcements, and grades. I'll do my best to let you know when something new and important comes up, but it is your responsibility to check the web site frequently for information that you might not get otherwise.

Exams
Exam material will come from the lecture notes, quizzes, book, programming assignments and labs. There will be more information about each exam as it approaches. The tentative exam dates are listed below. All exams will be taken during the regular class period. These *may change*, so as the date approaches make sure you've got the most recent information.

Exam 1: Friday, February 15th, 2013
Exam 2: Wednesday, March 13th, 2012
Exam 3: Friday, April 12th, 2012
Final: Wednesday, May 8th, 2012

If you are unable to take a scheduled exam, it may be possible to take a make-up exam provided that you do both of the following, which are then subject to my approval:

1. Make arrangements prior to the scheduled exam (for last minute emergencies, telephone me at 424-7080 or leave a message at the computer science office, 424-2068 or send me a text at the number on the first page). No after-the-fact notifications will be accepted.
2. Have a written medical excuse signed by the attending physician OR have a note of justification from the Dean of Students Office.

If allowed, only one make-up exam will be given. It will be a comprehensive exam given at an arranged time during the last week of the semester.

Quizzes
You will have quizzes throughout the semester. Quizzes are generally short and should only take a few minutes to complete. You will be given a quiz every 3-5 class periods to ensure you are staying current with the material. Your lowest quiz will be dropped. There are no make-up quizzes.

Grading
Course grades will be based on programming assignments, labs, quizzes and four exams. Your final grade will be computed as follows:

45% - programming projects
10% - quizzes
45% - exams

Grading will be on a plus/minus system. Grading *may* be done on a curve depending on the overall performance of the class. If no curve is used, your grade will be computed based on the following:

Percentage	Grade	Percentage	Grade
>91	A	71 - 77	C
89 - 91	A-	69 - 71	C-
87 - 89	B+	67 - 69	D+
81 - 87	B	61 - 67	D
79 - 81	B-	55 - 61	D-
77 - 79	C+	<55	F

Learning Outcomes
- Given an English description of a formal language, the student will be able to construct a context-free grammar that generates this language.
- Given a context-free grammar and the source code for a program, the student will be able to parse the program according to the grammar, and to produce a parse tree of the program or identify syntactic errors in the program.
- Given a context-free grammar in Backus-Naur Form (BNF), the student will be able to convert it to an equivalent, more compact grammar in Extended BNF.
- Given a context-free grammar, the student will be able to determine whether the grammar is ambiguous or not.
- Given an ML expression, the student will be able to deduce the type signature that the ML interpreter would infer for the expression.
- Given a program and a scoping mechanism (static or dynamic), the student will be able to trace the execution and infer the output of the program.
- Given a functional description of an operation, the student will be able to implement it as an ML function using either recursion or a computational pattern such as filtering, mapping, or folding, or a combination thereof.
- Given an ML "datatype" for a data structure, the student will be able to write ML expressions that create values for this data type or that implement operations on the data structure.
- 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
- λ-calculus
- Interpreters
- Evaluation order and parameter passing
- Object-orientation
- Parallel programming

Assignments

Most projects will consist of short programming projects. One of your goals should be to write understandable, readable code. You should be making every effort to comment anything that might be confusing to a reader unfamiliar with your program, to name variables intelligently, to use indentation that reflects the code's organization, and so on. All of this will be taken into account during grading: poorly organized or written code may have a negative impact on your grade, even if the resulting program works fine.

All assignments will need to be submitted electronically via d2l. It is your responsibility to ensure that all projects are submitted correctly. **Late assignments are penalized at 10% per day up to 7 days.** If you believe an assignment or exam was graded unfairly and would like to have it re-graded, please let me know about it *within one week* of having the assignment or exam returned to you. I will re-grade the entire assignment and you may gain or lose points.

Academic Dishonesty

Academic dishonesty of any kind will not be tolerated. All assignments, labs, quizzes and exams are to be completed individually. While discussion of ideas and problems with fellow students is encouraged, all projects and labs must be done individually. In certain circumstances, code fragments from the instructor may be provided to eliminate tedious coding or to provide a common framework for all students. **All other code must be original**. Online resources may be used to help you understand the material, but you may not copy online code nor can you "borrow" code from other students, past or present.

Any suspected academic dishonesty will be dealt with on a case-by-case basis. Any clarification of what does or does not constitute academic dishonesty must take place *before* you turn in questionable work. For clarification on what constitutes academic dishonesty, contact me or consult the printed policy in the UWO Student Discipline Code, Chapter UWS 14.