Compilers

Computer Science 431

Instructor:	Erik Krohn
E-mail:	krohne@uwosh.edu
Text Message Only:	608-492-1106
Class Time:	Tuesday & Thursday: 9:40am - 11:10am
Classroom:	Halsey 237
Office Location:	Halsey 216
Office Hours:	Monday: 12:20pm - 1:50pm Tuesday: 11:10am - 12:40pm Thursday: 12:30pm - 1:30pm
Prerequisites:	Math212/CS212 each with a grade of C or better and completion of or concurrent enrollment in CS331.
Course Website:	`http://www.uwosh.edu/d2l`
Number of Credits:	3

Course Information

An introduction to compiler writing techniques for translating a higher level programming source language into a lower level target language. Topics to be covered include: definition of programming languages, lexical and syntactic analysis, low level code generation and optimization, run time systems, and error detection, reporting, and recovery. A major programming project will be assigned to provide experience with compiler design.

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.

Mini Assignments

You will have daily mini assignments. Mini assignments are generally short and should take less than an hour to complete. You will be assigned a mini assignment every lecture to ensure you are staying current with the material. Mini assignments must be completed in LaTeX and the resulting pdf uploaded to the dropbox before the due date. Not all mini assignments will be graded. **No late mini assignments will be accepted.**

Assignments

All assignments requiring "written" work must be written in LaTeX and the resulting pdf submitted electronically via d2l. It is your responsibility to ensure your submission was submitted correctly. **There are no late submissions**.

Exams

Exam material will come from the lecture notes, mini assignments, book and assignments. 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 One** - Thursday, March 1^{st}, 2018
- **Exam Two** - Thursday, April 5^{th}, 2018
- **Exam Three** - Thursday, May 10^{th}, 2018

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 message. 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.

Grading

Course grades will be based on assignments, mini assignments and exams. Your final grade will be computed with the following percentages:

- 45% - assignments
- 15% - mini assignments
- 40% - exams

If you believe anything was graded incorrectly or unfairly and would like to have it regraded, you must let me know about it within *one week* of having the item graded. I will regrade the entire assignment or exam and you may gain or lose points.
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
≥ 92	A	72 - 78	C
90 - 92	A-	70 - 72	C-
88 - 90	B+	68 - 70	D+
82 - 88	B	62 - 68	D
80 - 82	B-	60 - 62	D-
78 - 80	C+	< 60	F

Topic Coverage

- Syntactic analysis
  - Lexical analysis
  - Using scanner and parser generators
  - Context-free grammars and parsing
  - Top-down parsing: Recursive-descent (or predictive)
  - Bottom-up parsing: LR(0), SLR(1), LR(1), LALR(1)
  - ASTs: Abstract syntax trees
- Semantic analysis
  - Symbol tables
  - Type checking
- Runtime organization
  - Activation records and run-time stacks
  - Runtime system (the JVM or other target architecture)
- Code generation
  - Java bytecodes or other assembly/machine language
  - Instruction selection
- As time permits, one or more of the following topics:
  - Liveness analysis
  - Register allocation
  - Garbage collection
  - Loop optimizations
  - Pipelining and scheduling
  - The memory hierarchy

Learning Outcomes

- Describe the compilation process, including the roles of the scanner (lexical analyzer). parser, semantic analyzer, intermediate code generator, optimizer, and code generator
- Define a deterministic finite state automaton (DFA)
- Explain how a DFA can be used to recognize a token
- Discuss the relationship between DFAs and regular expressions
- Define the lexical structure of a programming language using regular expressions
- Implement a lexical analyzer using a scanner generator
- Explain how a context-free grammar can be used to express the syntactical structure of a programming language
- Compare the respective capabilities of regular expressions and context-free grammars
- Construct a parse tree for an expression defined by a context-free grammar
- Discuss the differences between LL, SLR, LR and LALR grammars
- Trace the actions of a recursive descent parser in processing an expression defined by an LL grammar
- Trace the actions of a top-down table-driven predictive parser defined by an LL grammar
- Trace the actions of a bottom-up table-driven parser defined by an SLR/LR/LALR grammar
- Define the syntactical structure of a programming language using a context-free grammar
- Implement a parser using a parser generator
- Explain how syntax-directed translation is used to augment a context-free grammar with translation rules
- Define an abstract syntax tree
- Compare abstract syntax trees to parse trees
- Apply syntax-directed translation to be able to generate abstract syntax trees
- Explain the role of semantic analysis and its relationship to type checking and scoping issues in programming languages

- Discuss the role played by symbol tables in semantic analysis
- Compare different implementation strategies for symbol tables
- Implement a semantic analyzer for a programming language
- Explain what an activation record (stack frame) is and discuss its role in function calls and parameter passing
- Explain the role played by intermediate code generation and compare a compilation process that uses it to one that does not
- Implement a code generator for a programming language
- Test each of the implementations above with an appropriately designed set of test cases
- Apply sound software engineering principles, design patterns, and robust coding techniques, to successfully complete a large, software project

Academic Dishonesty

Academic dishonesty of any kind will not be tolerated. All assignments, mini-assignments and exams are to be completed individually unless otherwise specified. While high-level discussion of ideas and problems with fellow students is encouraged, all work 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. If you use a book, website or any reference to help you solve a problem, you must cite the reference in your assignment.

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.