CS 431 – Compilers – 3 Credits
Fall 2019

Instructor: David Furcy
Email: furcyd@uwosh.edu
Office: Halsey 221

Office Hours: **MWF** 10:10-11:30, **TR** 12:30-1:30, or **by appointment**

Class Meetings: **TR** 9:40-11:10 in HS 202
Prerequisites: A grade of C or better in CS 212 and completion of or concurrent registration in CS 331
Course Web Page: canvas.uwosh.edu
You should check this Canvas site daily.
Textbook: None required
References: Handouts will be made available on the course web page before each class. Print them and take ***detailed*** notes during each class. Fully annotated handouts will be your most useful reference for this course.

Tests: Exam #1: Week of October 28th
Exam #2: Week of December 9th

Description: 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.

Topic Coverage:

- Syntactic analysis
  - Lexical analysis
  - Using scanner and parser generators
  - Context-free grammars and parsing
  - Top-down parsing: Recursive-descent / predictive
  - Bottom-up parsing: LR (if time permits)
  - ASTs: Abstract syntax (or parse) trees
- Semantic analysis
  - Symbol tables
  - Type checking
- Runtime organization
  - Activation records and run-time stacks
  - The JVM
- Code generation
  - Translating ASTs into intermediate representation trees
  - Java bytecodes
  - Instruction selection

- If 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: At the conclusion of the course, the student will be able to:

- 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, 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 LR 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
- Collaborate with team members, apply sound software engineering principles, design patterns, and robust coding techniques, to successfully complete a large, software project

Course Grading Policy: Your final grade for this course will depend on a semester-long programming project, daily quizzes, and two exams. The goal of the project is to write a compiler for (a subset of) the Java language. Like *javac*, your compiler will translate Java programs into bytecodes to be executed by the Java Virtual Machine. The project will be broken down into about five assignments. Each assignment will be graded on a scale from 0 to 100. Each assignment will carry a weight between 1 and 5, with later assignments typically carrying a larger weight. Quizzes will be equally weighted. Your overall numerical grade for the course will be computed as the weighted sum of the component grades using the following weights:

Component	Weight
Overall project	35%
Quizzes	15%
Exam #1	25%
Exam #2	25%

Finally, your letter grade for the course will be computed using the following mapping:

Numerical Score	Course Grade	Numerical Score	Course Grade
≥ 92	A	≥ 72	C
≥ 90	A-	≥ 70	C-
≥ 88	B+	≥ 68	D+
≥ 82	B	≥ 62	D
≥ 80	B-	≥ 60	D-
≥ 78	C+	< 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 contact me *as soon as possible*. Do not wait until the end of the semester to bring up grading issues.

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 programming assignments on time.

It is hard to imagine how a student could do well in this course while missing classes, attending them unprepared, or not participating.

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 exam or the due date of an assignment, since you may not be the only student needing help at the last minute.

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 is considered late, **with no exception**. The penalty for late submissions is computed as follows:

Turned in	Penalty
on due date, after deadline	10%
one day late	30%
two days late	60%
three days late	100%

Note that submissions that are more than 2 days late will receive zero points. Late submissions can easily be avoided by starting to work on the assignment right away and asking me questions early if you get stuck.

The penalty for late submissions can be waived in **only one** scenario, namely if you give me a signed note from the attending physician or a written justification for the extension from the Dean of Students Office. If you miss a scheduled exam, you **may** be able to take a make-up exam provided you give me 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. If you miss a quiz, you **may** be able to take a make-up quiz, provided you have a valid justification (in writing) for your absence.

Collaboration versus Cheating: All submitted assignments must be the work of all, and only, the students on the team. While it is acceptable to discuss the assignments with others, you must submit your own work. You may not look at or "borrow" any piece of code of any length from anyone else, unless you can live with a zero and the other potential academic sanctions of cheating. Check out the UWO Student Discipline Code (UWS 14) at https://www.uwosh.edu/stuaff/images/Chapter%20UWS%2014.pdf/view for details.

Accommodations: The University of Wisconsin Oshkosh supports the right of all enrolled students to a full and equal educational opportunity. It is the University's policy to provide reasonable accommodations to students who have documented disabilities that may affect their ability to participate in course activities or to meet course requirements.

Students are expected to inform instructors of the need for accommodations as soon as possible by presenting an Accommodation Plan from either the Accessibility Center, Project Success, or both. Reasonable accommodations for students with disabilities is a shared instructor and student responsibility.

The Accessibility Center is part of the Dean of Students Office and is located in 125 Dempsey Hall. For more information, email accessibilitycenter@uwosh.edu, call 920-424-3100, or visit the Accessibility Center Website.

Disclosure: Students are advised to see the following URL for disclosures about essential consumer protection items required by the Students Right to Know Act of 1990:

https://uwosh.edu/financialaid/consumer-information/

Final Note: I expect every committed and hardworking student to do well in this course. I am looking forward to a fun and rewarding semester together. Given my high standards, I could not be more satisfied than if everyone earned an A in this course.

Have fun this semester and good luck!