CS375 Compilers and Interpreters
Fall, 2018

Prerequisites:

CS330, MAT258

General Information:

Class Schedule:	Tuesday / Thursday 10:30am-11:50am
Classroom:	Carr
Professor:	Trevor Sundberg
Contact:	360-731-1161, tsundburg@digipen.edu
Class web page:	https://digipen.edu/~tsundburg/CS375/
Office Hours:	Tuesday, 2:00pm – 4:00pm (3rd Floor Engineering)

Description:

This course presents fundamental topics in the field of compiler construction. Topics covered in the course will help students understand and implement a compiler for a high-level programming language. The course will guide the students towards an in-depth understanding of compilation techniques and runtime implementation for a modern programming language.

Course Objectives and Learning Outcomes:

- Students will understand how to write tokenizers for complex languages
- Comprehension of ECMAScript regular expressions
- Understanding grammars and recursive descent parsers
- Application of parsers to build syntax trees and generate code from those trees
- Apply knowledge of abstract syntax trees and visitors to build an interpreter

Optional Textbooks:

Compilers: Principles, Techniques, & Tools (Second Edition)

Alfred V. Aho, Monica S. Lam, Ravi Sethi, Jeffrey D. Ullman

Addison-Wesley

ISBN-10: 0-321-48681-1

Language Implementation Patterns

Terence Parr

The Pragmatic Bookshelf

ISBN-13: 978-1-93435-645-6

Grading Policy:
There are 5 assignments totalling up to 100% of your grade (20% each)..

A	93 – 100
A-	90 – 92.9
B+	87 – 89.9
B	83 – 86.9
B-	80 – 82.9
C+	77 – 79.9
C	73 – 76.9
C-	70 – 72.9
D	60 – 69.9
F	< 60

Late Policy:
Assignments submitted late up to one week after the due date will take a 25% penalty (max 75%). Every day after that the assignment will decrease in max score until the two week mark, where assignments will be worth a max of 20%.

Instructor's Biography:
Bachelors of Science in Computer Science in Real Time Interactive Simulation (2010)
Monolith / WB Games – Guardians of Middle Earth / Shadow of Mordor (2010-2011)
Lead of Zero Engine / Author of Zilch programming language (2011-Present)
Sucker Punch Productions – Ghost of Tsushima (2016-2017)

Academic Integrity Policy:
Academic dishonesty in any form will not be tolerated in this course. Cheating, copying, plagiarizing, or any other form of academic dishonesty (including doing someone else's individual assignments) will result in, at the extreme minimum, a zero on the assignment in question, and could result in a failing grade in the course or even expulsion from DigiPen. Grading server runs statistical analysis over assignments to flag copying.

Disability Support Services:
If students have disabilities and will need formal accommodations in order to fully participate or effectively demonstrate learning in this class, they should contact the Disability Support Services Office at (425)629-5015 or dss@digipen.edu. The DSS Office welcomes the opportunity to meet with students to discuss how the accommodations will be implemented. Also, if you may need assistance in the event of an evacuation, please let the instructor know.

Outline and Tentative Dates: *(Schedule and content is subject to change)*
** Means the assignment is due that week*

Week	Dates	Topics	Assignment
1		Overview of parts of the compiler. Building a Deterministic Finite Automata (DFA) and tokenization algorithm.	Tokenizer
2		More DFAs. Regular expressions syntax and operators. Regular languages and limitations.	
3		Extended operators and regular expression practice. Introduction to grammars.	***Tokenizer****
4		Grammar recursion, associativity, and operator precedence. Left/right-most derivations and parse trees. Elimination of ambiguity and left-recursion.	Parser
5		Recursive descent parsing. Recognition, errors, and recovery. Accept / expect pattern and look-ahead sets.	
6		Practical implementation techniques for recursive descent. Overview of unique pointer.	***Parser****
7		Abstract Syntax Trees (AST) and creation from a recursive descent parser. Visitation patterns for an AST (heterogeneous vs homogeneous nodes).	AST & Visitor
8		Evaluation of an AST & expression tree (interpreter). Passes and module based compilers.	
9		Semantic analysis and type checking Symbol table. Literal types, binary and unary operators, function return types, etc.	***AST & Visitor****
10		Scope tree and resolution. L and R value expressions. Type inference, implicit conversion, promotion, tree transformations, and operator overloading.	Semantic Analyzer
11		Intermediate representation, three address opcodes, and static single assignment form. Virtual byte codes and implementation patterns.	
12		Visiting the AST for code generation. Stack machine vs register machine.	***Semantic Analyzer**** Interpreter
13		Constant folding, dead code elimination, basic optimizations.	
14		Platform ABI compatibility & dynamic linked libraries. Overview of LLVM. IDEs and auto-complete.	
15			***Interpreter****