University of Kentucky

Department of Computer Science

CS 441G Compilers for Algorithmic Languages

1. ***Course Number/Name***: CS 441G, Compilers for Algorithmic Languages

2. ***Credits and Contact Hours***: 3 credits, 3 contact hours

3. ***Instructor***: Assigned by the department

4. ***Textbook***: *Implementing Programming Languages. an Introduction to Compilers and Interpreters (Texts in Computing Volume 16 )by Aarne Ranta.*

*(recommended) Basics of Compiler Design, Torben Mogensen; available in http://www.diku.dk/hjemmesider/ansatte/torbenm/Basics/basics\_lulu2.pdf*

5. a. ***Catalog Description***: The techniques of processing, specifying, and translating high-level computer languages are studied. Topics include finite state machines and lexical analysis, context-free grammars for language specification, attributed translation grammars, language parsing, and automatic generation of compilers by SLR, LALR, and other methods of analyzing context-free grammars. Other topics may include code optimization, semantics of programming languages, and top-down parsing.

b. ***Prerequisites***: Prereq: CS 315 and engineering standing.

c. ***Required course***:elective for CS, required for ECE

6. a. ***Outcomes of Instruction***:

Students will learn how a compiler for an algorithmic language is organized, designed, and constructed. Specifically, students will be able to demonstrate knowledge of and skills in how to:

1. specify lexicographical constructs describing elements of algorithmic languages;
2. specify parsing elements for algorithmic languages;
3. use regular expressions to simplify compiler generation;
4. design and implement a complete algorithmic language compiler;
5. use compiler generator tools such as lex (flex) and yacc (bison);
6. organize memory for both static and dynamic data types;
7. generate a parse tree that can be optimized before code generation;
8. generate translated code from the parse tree;
9. document a complex programming project, including documenting what works and what does not or is not yet implemented.

b. ***Contributions to Student Outcomes from Criterion 3 (Computer Science)***

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Outcome | a | b | c | d | e | f | g | h | i | j | k |
| CS 441G | 2 | 3 | 2 |  |  |  |  |  | 3 | 3 |  |

*3- Strongly supported 2 – Supported 1 – Minimally supported*

 7. ***List of Topics Covered:***

1. Organization and stages of the compilation process
2. Lexical analysis and regular expressions
3. Syntax-analysis, Context-Free Grammars, Parsing
4. Semantic Analysis and Type Systems
5. Intermediate Code Generation
6. Run-Time Memory Management
7. Code Generation and Optimization
8. Compiler-Compiler Tools