William Paterson University of New Jersey  
Department of Computer Science  
College of Science and Health  
Course Outline

I. **Course Title**: CS 420 Compiler Construction  
   3 credits

II. **Course Description**:
   An in-depth study of the principles and design aspects of programming language translation. The major components of a compiler are discussed: Lexical analysis, syntactic analysis, semantics routines and code generation. Alternative parsing strategies are presented and compared with respect to space and time tradeoffs.

III. **Prerequisites**: CS 382 with a grade of C- or better

IV. **Course Objectives**:
   1. To understand the structure and major functions of a compiler.
   2. To understand the fundamentals of compilation and compiler technology.

V. **Student Learning Outcomes**:
   After the completion of this course, a successful student will be able to do the following:
   1. Write and execute a scanner to recognize simple tokens
   2. Write and execute a recursive descent parser for a simple programming language
   3. Apply LL parsing techniques on a simple programming language
   4. Apply LR parsing techniques on a simple programming language.
   5. Apply the symbol table mechanism.
   6. Generate intermediate codes for a simple programming language.
   7. Explain run-time storage allocations.
   8. In addition, The course will also reinforce the following students learning outcomes of the university:
      a) Effectively express themselves in written and oral form. Measure: exams and projects.
      b) Demonstrate ability to think critically. Measure: exams and projects.
      c) Demonstrate ability to integrate knowledge and ideas in a coherent and meaningful manner. Measure: exams and projects.

Assessment tools for measuring the degree of achievement of goals 1-7 are: exams, surveys, and projects.
VI. **Course Contents:**

1. **Introduction:** Programming languages and language processors; types of compilers; structure of a compiler.
2. **Scanning:** Lexical elements of a programming language; regular expressions and regular sets; finite automata and scanners; scanner generators.
3. **Grammars and parsing:** Context-free grammars and extended BNF grammars; parsers and recognizers; grammar analysis algorithms.
4. **LL(1) grammars and parsers:** LL(1) predict function; LL(1) parse table; building recursive descent parsers from LL(1) tables; making grammars LL(1), parsers generators.
5. **LR parsing:** Shift-reduce parsers; LR parsers; SLR parsers; LALR parsers.
6. **Semantics processing:** Syntax-directed translation; semantics processing techniques and intermediate representations.
7. **Symbol tables:** Basic implementation techniques; attributes in symbol tables.
8. **Code generation and optimization:** Attribute grammars and action routines.
9. **Run time storage organization:** Static allocation; stack allocation; heap allocation; program layout in memory, static and dynamic chains.

VII. **Teaching Methods:**

1. Classroom lectures
2. Exercises/homework/lab assignments discussions
3. Open Lab. Sessions

VIII. **Evaluation:**

1. Periodic examinations and final examination.
2. Homework assignments.
3. Programming assignments: Students will be required to design and implement components of a compiler for a small but representative language.

IX. **Textbook:**


X. **Bibliography:**

“Modern Compiler Design”, 2005, David Galles, Addison Wesley.
“Advanced Compiler Design and Implementation”, 1997, Steven Muchnick, Morgan Kaufmann.

XI. Prepared by: Dr. Gilbert Ndjatou

XII. Original Department Approval Date: Spring 1997

XIII. Revised by: Dr. Gilbert Ndjatou on Spring 2005 and previously on April 1, 2000

XIV. Department Revision Approval Date: Spring 2005.