Course Title: COSC 4364: Compilers

Semester Credit Hours: 3 (3,0)

I. Course Overview

This course is the study of the theory and practice of constructing a compiler, including lexical analysis, parsing, semantic analysis, run-time organization, code generation, and optimization. During the course of the semester, the students complete a significant compiler project.

II. PMU Competencies and Learning Outcomes

Students in this course develop conceptual and programming skills necessary for continued success in computer science. The skills enhance their abilities to appreciate the theory and practices of compiler construction common to computer science as a discipline and to effectively communicate their solutions to fellow professionals. This course makes extensive use of the PMU technology infrastructure to provide communication between faculty and students. The course includes individual as well as group projects, establishes both conceptual reasoning skills and technical communication skills, and provides opportunities for the presentation and defense of designed solutions.

III. Detailed Course Description

COSC 4364 is concerned with the study of the theory and practice of constructing a compiler, including lexical analysis, parsing, semantic analysis, run-time organization, code generation, and optimization. During the course of the semester, the students complete a significant compiler project. The techniques studied in this course are particularly useful for programming applications that are the input of other programs. Thus, the course enriches students to formulate, design, and implement solutions to open programming problems.

IV. Requirements Fulfilled

COSC 4364: Compilers satisfies three hours of the requirements for the degree in computer science. It is required of all students pursuing a degree program in computer science within the College of Information Technology. Students are recommended to take the course in the second semester of the senior year.

V. Required Prerequisites

- COSC 3351: Algorithms
- COSC 4461: Programming Languages

VI. Learning Outcomes

In this course, students learn:

- To understand and apply the phases in compilation process.
- To understand the concept of lexical analysis, and apply the tools for lexical analysis.
- To acquire the concept of parsing, and apply the tools for parsing.
- To learn and apply the concept of semantic analysis.
- To understand the run-time environments in compilation.
- To acquire the knowledge of code generation and optimization.
- To develop improved communication and collaborative skills.

VII. Assessment Strategy

This course is designed with three primary goals in mind: to introduce students to the conceptual basis and practical issues associated with the compiler construction, to enhance students' programming techniques to its application in computer science, and to provide students with the opportunity to communicate their designs and implementations to their peers in a professional setting. With this in mind, the course grade involves an assessment of their performance on examinations that focus on the application of programming paradigms to the solutions of problems, the performance analysis of the designed solutions, and the communication of designed solutions to those problems to an audience. Course grades are based on:

- Weekly assigned homework to motivate students to do the work and earn credit accordingly.
- Weekly, in-class presentations by students of solutions to real world problems related to the course material and classroom discussion and critique of the presentation.
- One in-class examination to assess the student's accumulative mastery of content covered prior to the time of the examination.
- 5 programming assignments testing students understanding of the major concepts introduced during the course.
- A comprehensive final examination to assess the student's accumulative mastery of course material.

The final grade is based on 10% credit for the homework, 10% for the presentations and participation in classroom discussion, 10% on in-class examinations, 50% on programming assignments and 20% for the final examination.

Students are required to maintain a journal of thoughts and commentaries during the course. The journal contains daily entries including the identification of areas of interest and concern, notes on the preparation of presentation and comments and analysis of classmate's presentations. The journal is reviewed weekly by the instructor to provide feedback to the students.

Final grades and the student and instructor observations from reflective notebooks are included in the student's portfolio for use in the final assessment capstone course. The intent is to document the student's maturation as he or she proceeds through the curriculum.

VIII. Course Format

This course utilizes both lecture/discussion and laboratory exercises. Students are expected to attend three hours of lecture per week. At least once per week students should be prepared to make presentation on the design and implementation of a solution to a problem selected by the instructor and to take part in a discussion based on that presentation. Once a week students should have at least 30 minutes of collaborative problem solving activity.

Classroom Hours (3 hours per week) **Class: 3**
Lab: 0

Web supplement:

Course home page (the university's Web tool, WebCT or Blackboard) should contain the following:

- Course syllabus
- Course assignments
- Sample solutions to examinations (after being graded and returned)
- Sample solutions to programming assignments (after being graded and returned)
- Course calendar (an active utility)
- Course e-mail (an active utility)
- Course discussion list (an active utility)
- Student course performance (an active utility)

IX. Topics to be Covered

A. Phases in compilation process
B. Lexical analysis
    a. Regular expressions and finite state automata (FSAs)
    b. Non-deterministic finite state automata (NFAs)
    c. Subset construction
    d. Thompson's construction
C. Parsing
    a. Bottom-up parsing
    b. LR parsing algorithm
    c. Semantic actions
    d. Abstract syntax trees
    e. SLR parser
    f. Top-down parsing

D. Semantic analysis
   a. Scopes
   b. Symbol tables
   c. Type-checking
E. Run-time organization
   a. Scoping and storage allocation
   b. Activation records
   c. Non-Local references
   d. Parameter-passing techniques
F. Code generation
   a. Intermediate code generation
   b. Final code generation
G. Machine-independent optimizations

X. Laboratory Exercises

This course does not offer a separate laboratory to students.

XI. Technology Component

This course makes use of the university's wireless access infrastructure. The course relies on the university and the students having access to professional grade application development environments for the students to use.

XII. Special Projects/Activities

Students are required to keep a "reflective notebook" in which, after each class, they enter their own assessments of what they learned, and what questions remain from the class. From each exercise set, each student selected one problem, which the student thinks best reflects the way the topic is used in a technical context. A detailed solution to the problem is included in the student's reflective notebook.

XIII. Textbooks and Teaching Aids

A. Required Textbook

T. W. Parsons, *Introduction to Compiler Construction*, (1992) W. H. Freeman
ISBN 0-7167-8261-8

B. Alternative Textbooks

None

C. Supplemental Print Materials

None

D. Supplemental Online Materials

As available from publisher.