Course Title: COSC 4461: Programming Languages

Semester Credit Hours: 4 (3,1)

I. Course Overview

Programming languages is the study of basic concepts and constructs underlying the design of the modern programming languages. Various programming paradigms, including object-oriented, functional, logic, and concurrent programming, are discussed.

II. PMU Competencies and Learning Outcomes

Students in this course develop skills necessary for understanding the design of the modern programming languages so as to appreciate the strengths and limitations among different programming paradigms. These skills are necessary for continued success in computer science. This course makes extensive use of the PMU technology infrastructure to provide communication between faculty and students. The course includes a structured laboratory component to ensure that students gain the necessary experience and skill in managing the concepts introduced in the class. The course includes individual as well as group projects and provide opportunities for the presentation and defense of designed solutions.

III. Detailed Course Description

COSC 4461: Programming Languages is concerned with the study of basic concepts, including types, control structures, abstraction mechanisms, inheritance, concurrency, and constructs underlying the design of the modern programming languages. Various programming paradigms, including object-oriented, functional, logic, and concurrent programming, are discussed. Students are exposed to the implementation of programs using programming languages under different programming paradigms. One important lasting effect of this course is to enhance and develop the ability to design, implement and test solutions to effectively solve programming problems utilizing appropriate programming languages.

IV. Requirements Fulfilled

This course satisfies four 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. It should be taken in the first semester of the junior year.

V. Required Prerequisites

- GEIT 1412: Computer Science II
- COSC 3411: System Programming

VI. Learning Outcomes

In this course, students learn:

- To develop an understanding of various basic concepts and constructs underlying the design of the modern programming languages.
- To earn experience in designing and testing solutions to programming problems implemented with various programming paradigms.
- To be able to discuss the strengths and limitations of different programming paradigms in solving programming problems.
- 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 use of various programming paradigms to solve programming problems, to appreciate the strengths and limitations of different paradigms, 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 understanding of various concepts and constructs underlying the design of the modern programming languages, 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.
- Weekly structured laboratory exercises designed to guide students through specific course topics.
- Two in-class examinations 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, 20% for weekly laboratory exercises, 20% on in-class examinations, 30% on programming assignments and 10% 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 proceeds through the curriculum.

VIII. Course Format

A. Instruction

This course utilizes both lecture/discussion and laboratory exercises. Students are expected to attend three hours of lecture/discussion per week and two hours of laboratory 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.

B. 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)

Classroom Hours (5 hours per week) **Class:** 3
Lab: 2

IX. Topics to be Covered

A. Variety of programming languages
   A. Abstraction
   B. Compilers and interpreters
   C. Syntax and semantics
   D. Context-free grammars
B. Imperative languages
   A. Control structures
   B. Data types and their presentation
   C. Composite types
   D. Subprograms, functions, procedures, methods
   E. Program structure
   F. Exception handling
C. Programming paradigms
   A. Object-oriented programming
   B. Functional programming
   C. Logic programming
   D. Concurrent programming

X. Laboratory Exercises

This course requires a weekly 2-hour laboratory component. Topics to be covered in the laboratory sessions should include:

- Compilers – exercises in using various programming language compilers in UNIX.
- make Utility – exercises in using the UNIX make utility.
- LISP I – exercises in familiarizing the use of the functional programming language LISP.
- LISP II – additional exercises in familiarizing the use of LISP.
- Prolog I – exercises in familiarizing the use of the logic programming language Prolog.
- Prolog II – additional exercises in familiarizing the use of Prolog.
- Java I – exercises in familiarizing the use of Java.
- Java II – additional exercises in familiarizing the use of Java.
- Constructors and Destructors – exercises in using the constructors and destructors in C++.
- List Manipulation – exercises in performing list manipulations in LISP.
- Unification – exercises in performing unifications in Prolog.
- Concurrency – exercises in exploiting concurrency in Java.
- Three additional lab sessions should be kept in reserve to allow the instructor to extend the more difficult laboratories for more than one session.

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. The course has a laboratory component that would be best implemented in university provided laboratory space.

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 selects 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

Sethi, R.. *Programming Languages: Concepts and Constructs*. _____: Addison-Wesley,1995.
ISBN 0-201-59065-4

B. Alternative Textbooks

Krishnamurthi, S. *Programming Languages: Application and Interpretation*. _____: _____, 2003.
Available at:
http://www.cs.brown.edu/~sk/Publications/Books/ProgLangs

C. Supplemental Print Materials

None.

D. Supplemental Online Materials

As available from the publisher.