CSC 511 - PROGRAMMING LANGUAGES

CREDIT HOURS: 3
PREREQUISITES: Nine advanced hours of computer science (CSC 441 recommended)

CATALOG DESCRIPTION

Issues of programming language design including data abstraction, concurrency, exception handling, subprograms, data types, control structures, and describing syntax and semantics. Alternative paradigms such as imperative, functional, logic, and object-oriented.

PURPOSE OF COURSE

To acquaint students with criteria appropriate for the critical evaluation of existing and future programming languages and constructs.

EDUCATIONAL OBJECTIVES

Upon successful completion of the course, students should be able to:

1. Identify issues of programming language design including data abstraction, concurrency, exception handling, subprograms, data types, control structures, syntax, and semantics.
2. Identify criteria appropriate for the critical evaluation of existing and future programming languages and constructs.
3 Describe PL design principles.
4. Describe aspects of the imperative programming language paradigm in depth.
5. Compare several object-oriented programming languages.
6. Identify object-oriented programming language design issues.
7. Describe functional programming language design concepts and describe at least one previously-unfamiliar functional programming language.
8. Demonstrate familiarity with logic programming language design concepts and with a logic programming language.
9. Demonstrate the ability to learn a new programming language without help from anyone else, develop illustrative programs in the programming language, produce a tutorial for the language, and present the tutorial to the instructor and the other students in a multimedia classroom setting.

APPROACH

Design issues for the primary constructs of the languages are explored and examples from a variety of languages are presented. Imperative languages are studied by using extensive textual material and accompanying examples. Object-oriented languages are examined through the use of text and instructor exposition as well as through student research and presentation. Functional and logic programming languages are studied by means of description and examples.

CONTENT **Hours**

Introductory Concepts of Programming Language Design 6
- Programming Domains
- Evaluation Criteria
- Design: Influences and Compromises
- Implementation Methods
- Evolution of the Major Programming Languages
- Describing Syntax and Semantics

Imperative Programming Language Constructs and Design Issues 12
- Names, Bindings, Type Checking, and Scope
- Data Types: Primitive, Structured, Pointer
- Expressions and the Assignment Statement
- Statement-Level Control Structures
- Subprograms
- Abstract Data Types
- Concurrency
- Exception Handling

Object-Oriented Programming Language Exposition and Design Issues 9

Functional Programming Language Design Concepts 6

Logic Programming Language Design Concepts 6

Term Paper Presentations 3

Exams (plus final) 3

TOTAL 45

REFERENCES

Cezzar, Ruknet, A Guide to Programming Languages, Artech House, 1995.

Finkel, Advanced Programming Language Design, Addison-Wesley, 1996.

Friedman, L., Comparative Programming Languages, Prentice-Hall, 1991.

Horowitz, E., Fundamentals of Programming Languages, 2nd Ed., Computer Science Press, 1984.

Horowitz, E., Programming Languages: A Grand Tour, 3rd Ed., Computer Science Press, 1987.

Sebesta, Robert W., Concepts of Programming Languages, 7th Ed., Addison-Wesley, 2005.

Scott, Michael L., Programming Language Pragmatics, 2nd Ed., Morgan Kaufmann, 2006.