Course Number: CSC 600
Course Title: Programming Language Design
Number of Credits: 3
Schedule: Three hours of lecture/discussion per week.
Prerequisite: a grade of C or better in CSC 413
Catalog Description:
Design concepts for high-level programming languages. Comparative study of several languages and an introduction to grammars and parsing techniques. Extra fee required.

Expanded Description:

Syllabus:

1. *Survey, Classification, History, and Concepts of Programming Languages.* Imperative, object-oriented, logic, and functional languages: concepts and applications. Procedural and non-procedural programming. Syntax and semantics of programming languages. Evaluation and comparison of programming languages. Systematic presentation of programming language concepts. Selecting languages for specific applications.
2. *Presentation/survey and comparison of specific languages:* Prolog, Scheme, Ruby, Fortran, Cobol, Basic, Pascal and other languages. Program development in selected languages: procedural programming (**C, C++**), logic programming (**Prolog**), functional programming (**Scheme**), and OO programming (**Ruby**). Translation of programs from a source language to a given target language. Reuse of legacy programs and program libraries. Software performance issues.
3. *Design of languages and language processors***.** Macro processors and language expansion techniques. Design and implementation of interpreters and virtual machines.

Instructor: Dr. Jozo J. Dujmović (English spelling/pronunciation: Yozo Dooy-mo-vich)
Office: Thornton 946, Tel. (415) 338-2207,
Web site: http://cs.sfsu.edu/People/jozo/jozo.html. E-mail: jozo@sfsu.edu
Office hours: MWF 1-2pm.

Recommended literature:

1. J.J. Dujmović, *Programming Languages*. J. Wiley, 2003.
2. A.B. Webber, *Modern Programming Languages: A Practical Introduction*. Franklin, **:** Beedle & Asso., 2002. http://www.webber-labs.com/mpl.html (downloadable ppt lectures)
3. T.W. Pratt and M.V. Zelkowitz, *Programming Languages Design and Implementation.* Fourth Edition. Prentice Hall, 2001.
4. R. Sebesta, *Concepts of Programming Languages*. Fifth Edition. Addison Wesley, 2002.
5. G. Springer and D.P. Friedman, *Scheme and the Art of Programming.* The MIT Press and McGraw-Hill, 1989.
6. D. Flanagan and Y. Matsumoto, *The Ruby Programming Language.* O'Reilly, 2008.
7. Free books: http://computing.unn.ac.uk/staff/cgpb4/prologbook/
   http://rsusu1.rnd.runnet.ru/develop/fortran/prof77/prof77.html
   http://www.schemers.org/Documents/Standards/R5RS/r5rs.pdf
   http://www.freeprogrammingresources.com/ruby-tutorial.html

On-line: iLearn (http://ilearn.sfsu.edu/) – all communication with the class and distribution of materials will be on iLearn
Final Exam The final exam will be scheduled according to the University Calendar.
Attendance The attendance in classes is **mandatory** and will be periodically checked by taking the class roll.

Notes Take notes in the class. Class notes are extremely important for preparing for exams.

Effort Approximately 1-2 hours of work every day during the semester.

Grading: The total of 100 points is distributed as follows: *programming assignments* (20), *midterm exam* (30), and *final exam* (50). The total number of attained points is used for relative ranking of students. Letter grades are assigned taking into account three components: (1) the total score, (2) the relative ranking, and (3) the attendance of lectures and the class/iLearn activity.

Course Objectives and Role in Program

CSC 600 is the last course in the sequence of mandatory software courses 210-213-313-413-600. Since 210, 213, 313, and 413 are devoted only to object-oriented languages (C++ and Java), the objectives of CSC600 are to uniformly present all categories of languages, focusing on procedural, nonprocedural, logic, functional, and OO programming. Consequently, the objectives of this course include:

- Develop detailed understanding of general concepts of programming languages.
- Expose students to general language-independent algorithmic thinking and software development.
- Promote performance awareness at all levels of program development.
- Develop understanding for procedural and nonprocedural aspects of programming.
- Present the history of programming languages and a spectrum of actual languages including Fortran, Pascal, Basic, Cobol, Prolog, Scheme, and Ruby.
- Develop practical programming skills in procedural, nonprocedural, logic, functional, and object-oriented programming, exemplified with C, Prolog, Scheme, and Ruby.
- Highlight outstanding features, and review programming in other languages (e.g. Fortran, Pascal, Basic, Cobol, and scripting languages)
- Develop respect for language standards, and cooperative spirit of language communities.
- Provide background for advanced work in a graduate program.

Learning Outcomes

At the end of this course students will

- Approach software development using good programming concepts distributed in a spectrum of languages, as opposed to limit their activities to the frame imposed by their native programming language.
- Be able to write effective (correct and efficient) procedural code to solve small to medium sized problems. In particular, students will be able to identify programming components that contribute to various aspects of performance of software products (algorithmic performance, speed, low memory consumption, reduction of complexity, and reduction of development effort).
- Be able to read and understand selected legacy software, translate it to modern languages, and reuse.
- Have comparative programming experiences in procedural, nonprocedural, functional, logic, and OO programming obtained through programming in C, Scheme, Prolog, and Ruby.
- Be prepared to understand and evaluate the quality of software produced by other programmers, as a prerequisite for efficient management of programming teams.
- Make educated selection of programming languages, and use multiple languages in the development of software products.