### Course Change Proposal Form A

<table>
<thead>
<tr>
<th>Academic Group (College):</th>
<th>Academic Organization (Department):</th>
<th>Date: September 2, 2011</th>
</tr>
</thead>
<tbody>
<tr>
<td>ECS</td>
<td>Computer Science</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Type of Course Proposal:</th>
<th>Department Chair: Cui Zhang</th>
<th>Submitted by: Cui Zhang</th>
</tr>
</thead>
<tbody>
<tr>
<td>New X Change Deletion</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Does this course fulfill a requirement for single-subject or multiple subject credential students?</th>
<th>For Catalog Copy:</th>
<th>CCE (Extension):</th>
<th>Semester Effective:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes X No X</td>
<td>Yes X No</td>
<td>Yes X No</td>
<td>Fall Spring X, 2012</td>
</tr>
</tbody>
</table>

This course replaces experimental course Subject Area (prefix) and Catalog Nbr (course number):

If changing an existing course, should new version be considered a repeat of the original version? If so, the same Course ID will be maintained. If not, a new Course ID will be assigned. Note: In PeopleSoft terminology, the Course ID is the unique system identifier, not the Catalog Nbr.

Yes No

<table>
<thead>
<tr>
<th>Change from:</th>
<th>Title:</th>
<th>Units:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject Area (prefix &amp; Catalog Nbr (course no.):</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Change to:</th>
<th>Units:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subject Area (prefix &amp; Catalog Nbr (course no.):</td>
<td>Title:</td>
</tr>
<tr>
<td>CSC 135</td>
<td>Computing Theory and Programming Languages</td>
</tr>
</tbody>
</table>

**JUSTIFICATION:**

It is a necessary element in the computer science department's introduction of concentrations. It is a reduction in the number of required courses that computer science majors take. Analysis of ACM/IEEE curricular recommendations and of what is needed by later courses has guided the merging of essential material in formerly required courses CSC 132 and CSC 136 into this single newly required course.

**NEW COURSE DESCRIPTION:** (Not to exceed 80 words, and language should conform to catalog copy. See http://www.csus.edu/umanual/acad.htm - Guidelines for Catalog Course Description)

Introduction to computing theory with examples and applications. Automata and formal languages; regular expressions; deterministic and non-deterministic finite automata; pumping lemma for regular languages; push-down automata and context-free grammars; language recognition; parsing techniques including recursive descent; Turing machines; computable and non-computable functions. Design and implementation of selected features of programming languages. Functional and logic programming paradigms.

**Note:**

Prerequisite: CSc 28, CSc 35, CSc 130  
Enforced at Registration: Yes X No  
Corequisite:  
Enforced at Registration: Yes No  
Graded: Letter X Credit/No Credit  
Instructor Approval Required? Yes No X  
Course Classification (e.g., lecture, lab, seminar, discussion): Lecture  
Title for CMS (not more than 30 characters): COMP THEORY & PROG LANGS  
Cross Listed? Yes No X  
If yes, do they meet together and fulfill the same requirement, and what is the other course.  
How Many Times Can This Course be Taken for Credit? 1  
Can the course be taken for Credit more than once during the same term? Yes No X
FOR NEW COURSE PROPOSALS OR SUBSTANTIVE CHANGES ONLY:

**Description of the Expected Learning Outcomes:** Describe outcomes using the following format: “Students will be able to: 1), 2), etc.” See the example at http://www.csus.edu/acaf/example.htm

Students will be able to...

1. Design generators and recognizers for regular and context-free languages.
2. Illustrate the use of grammars, parse trees and syntax diagrams.
3. Explain the limits of regular and context-free languages and how to show a language is not regular.
4. Describe the connection between generators and recognizers.
5. Discuss parsing and scanning techniques including principle of longest substring, recursive-descent or table-driven parsing, derivations, parse trees, first and follow sets.
6. Explain programming language features and their use, including parameter-passing methods, dynamic and static scoping, static vs. dynamic binding, lifetime and associated performance implications, higher-order functions, lambda functions, tail-recursion, resolution and unification.
7. Use functional and logic programming languages to solve simple problems.
8. Describe the importance of Turing machines and the halting problem.

**Attach a list of the required/recommended course readings and activities [Note: it is understood that these are updated and modified as needed by the instructor(s).] This attachment should be forwarded only to your Dean's office, not Academic Affairs.

**Assessment Strategies:** A description of the assessment strategies (e.g., portfolios, examinations, performances, pre- and post-tests, conferences with students, student papers) which will be used by the instructor to determine the extent to which students have achieved the learning outcomes noted above:

**Examination questions, lab projects**

**For whom is this course being developed?**
Majors in the Dept. X, Majors of other Depts. ____, Minors in the Dept. ____, General Education ____, Other ____

Is this course required in a degree program (major, minor, graduate degree, certificate)? Yes X, No ____

If yes, identify program(s): BS in Computer Science

Does the proposed change or addition cause a significant increase in the use of College or University resources (lab room, computer facilities, faculty, etc.)? Yes ____ No X ____

If yes, attach a description of resources needed and verify that resources are available.

Indicate which department or programs will be affected by the proposed course (if any).

**The Department Chair's signature below indicates that affected programs have been sent a copy of this proposal form.**

**Accessibility:** Following course approval, and prior to the start of the semester in which the new or revised course will be taught for the first time, an accessibility checklist [available at http://www.csus.edu/accessibility/checklist.html] shall be completed and submitted to the appropriate Dean's office. An accessible syllabus shall also be made available online, preferably prior to the start of that semester's open registration period.

**Approvals:** If proposed change, new course or deletion is approved, sign and date below. If not approved, forward without signing to the next reviewing authority, and attach an explanatory memorandum to the original copy.

**Signatures:**
Department Chair: [Signature] 9/2/2011
College Dean or Associate Dean: [Signature] 9/19/11
CPSP (for school personnel courses ONLY)
Associate Vice President and Dean for Academic Programs

**Distribution:** Academic Affairs (original), Department Chair and College Dean. Dean's office to send original after approval to Academic Affairs, at mail zip 6016. An electronic copy must also be sent.

5/20/2010
COURSE DESCRIPTION

Dept., Number    CSC 135    Course Title    Computing Theory and Programming Languages
Semester hours    3    Course Coordinator    Scott Gordon
Course URL

Catalog Description

Introduction to computing theory with examples and applications. Automata and formal languages; regular expressions; deterministic and non-deterministic finite automata; pumping lemma for regular languages; push-down automata and context-free grammars; language recognition; parsing techniques including recursive descent; Turing machines; computable and non-computable functions. Design and implementation of selected features of programming languages. Functional and logic programming paradigms. Prerequisite: CSe 28, CSe 35, CSe 130

Textbook


References


Course Goals

To provide students with:

1. familiarity with the theoretical foundations of Computer Science.
2. facility with the concepts, notations, and techniques of the theories of automata, formal languages, and Turing machines.
3. understanding of selected programming language features and their implementation.
4. experience using and implementing a recursive-descent parser.
5. experience writing programs using functional and logic language paradigms.

Prerequisites by Topic

Thorough understanding of:
- At least one higher-level programming language.
- Sets and basic operations on sets.

Basic Understanding of:
- Fundamental concepts of data structures and algorithms, recursion.
- Basic operations on sets: subset, intersection, union, Cartesian product.
• Functions, their definition and properties.
• Regular expressions (can compose a valid expression for a simple set of strings).
• Finite automata (is able to construct a valid recognizer for a simple set of strings).

**Major Topics Covered in the Course**

1. Regular expressions, deterministic and non-deterministic finite automata (5 hours).
2. Language specification, syntax and semantics, compilation vs. interpretation, and programming language translation process (3 hours).
3. Regular languages and scanning (3 hours).
4. Nonregular languages and the pumping lemma (3 hours).
5. Context free grammars and languages, BNF/EBNF, and Chomsky’s hierarchy (6 hours).
6. Push-down automata and related parsing techniques, implementation of recursive-descent parsing (6 hours).
7. Turing machines and computability (3 hours).
8. Subprograms, recursion and parameter passing (2 hours).
9. Declarations, scope, lifetime, binding, and storage management (2 hours).
10. Functional programming languages, higher-order and lambda functions (6 hours).
11. Logic programming languages, resolution and unification (6 hours).

**Outcomes**

*Thorough understanding of:*
• Regular expressions: can develop an expression for a regular language.
• Finite state automata: can design an automaton to accept a regular language.
• Formal description of programming language syntax: BNF and EBNF grammars, parse trees, and syntax diagrams.

*Basic Understanding of:*
• Regular languages: their limitation, relation with regular expressions and finite automata.
• Relation between languages, grammars, and automata.
• Context-free grammars: can develop a CFG for a given language, understands the power and limitation of CFG.
• Push down automata: can design a PDA to accept a context-free language.
• Relationship between CFGs, CFLs, and PDAs.
• Parsing and scanning techniques: principle of longest substring, recursive-descent or table-driven parsing, derivations, parse trees, first and follow sets.
• Fundamental concepts of the major programming paradigms and characteristics.
• Language features, including parameter-passing methods, dynamic and static scoping, static vs. dynamic binding, lifetime, and associated performance implications.
• Functional programming and functional languages such as Scheme, LISP, or ML, higher-order functions, lambda functions, and tail-recursion.
• Logic programming: Prolog, resolution and unification, program design methods.
Exposure to:
- Turing machines and the halting problem.

Laboratory projects

1. Regular expressions and finite automata (2 weeks)
2. Pumping lemma (2 weeks)
3. Push-down automata (2 weeks)
4. Recursive descent parser (2 weeks)
5. Turing machines (2 weeks)
6. Scheme (2 weeks)
7. Prolog (2 weeks)

Estimated Curriculum Category Content (Semester hours)

<table>
<thead>
<tr>
<th>Area</th>
<th>Core</th>
<th>Advanced</th>
<th>Area</th>
<th>Core</th>
<th>Advanced</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithms</td>
<td></td>
<td>.5</td>
<td>Data Structures</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software Design</td>
<td></td>
<td></td>
<td>Concepts of Prog. Languages</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>Comp. Org. &amp; Arch.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Oral and Written Communications

none

Social and Ethical Issues

none

Theoretical Content

Half of the course consists of theoretical content: formal languages and abstract machines, computability, programming paradigms.

Problem Analysis and Solution Design

Most computer science students are interested in practical matters, and their strength lies in their ability in problem solving that is necessary for computer programming. Because of this, this course emphasizes problem solving. Students learn the material primarily through problem-type illustrative examples that show the motivation behind the concepts, as well as their connection to the theorems and definitions. Homework and in-class exercises contribute to a major part of the learning process. As students do programming exercises they are expected to analyze the problem and design a solution.
### Relationship between Course Outcomes and Program Outcomes

<table>
<thead>
<tr>
<th>Program Outcome</th>
<th>Mapping from Course Outcome to Program Outcome</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>Students will be able to apply mathematical concepts to develop formal descriptions of various machines. They will become familiar with formal methods of representation of the syntax and semantics of programming languages. They will understand the relation between parsing techniques and formal grammars. Students will use appropriate languages to write programs to solve logic problems. They will understand performance issues associated with different paradigms.</td>
</tr>
<tr>
<td>b</td>
<td>Students will be able to analyze a given problem, and evaluate the appropriateness of different paradigms to devise a solution, and compare various alternatives to select appropriate data structures and algorithms for its solution depending on the paradigm being employed.</td>
</tr>
<tr>
<td>c</td>
<td>Students will apply design principles to construct software solutions to given programming assignments</td>
</tr>
<tr>
<td>d</td>
<td>Students will be able to use a standard parsing technique. They will be able to use a number of different programming languages and will understand the design principles behind some of the features of programming languages.</td>
</tr>
<tr>
<td>i</td>
<td>Students will develop an understanding of programming languages and features that will allow them to adapt to future changes in the technology.</td>
</tr>
</tbody>
</table>

### Assessment Plan for the Course

- Exam questions.

Status: Approved by Curriculum Committee: 2/28/2011  
Approved by Department: 5/9/2011