### Course Change Proposal

**Form A**

<table>
<thead>
<tr>
<th>Academic Group (College):</th>
<th>Academic Organization (Department):</th>
<th>Date:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Engineering and Computer Science</td>
<td>Computer Science</td>
<td>1/26/15</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Does this course fulfill a requirement for single-subject or multiple subject credential students?</th>
<th>Yes <em>x</em> No <em>x</em></th>
<th>For Catalog Copy:</th>
<th>Yes <em>x</em> No <em>x</em></th>
</tr>
</thead>
<tbody>
<tr>
<td>CCE (Extension):</td>
<td>Yes <em>x</em> No <em>x</em></td>
<td>Semester Effective:</td>
<td>Fall <em>x</em> Spring <em>x</em> 2015</td>
</tr>
</tbody>
</table>

This course replaces experimental course Subject Area (prefix) and Catalog Nbr (course number): Yes _x_ No _x_.

**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.**

| Change from: | | |
|--------------| | Units: |
| Subject Area (prefix) & Catalog Nbr (course no.): | Title: | |

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

**JUSTIFICATION:**

In order to be consistent with recent changes to the Computer Science bachelor's program, specifically the approved merge of CSC 132 and CSC 136 into CSC 135, CSC 151 must update the prerequisites. Previously the prerequisites were "CSC 135 or CSC 136; CSC 136 may be taken concurrently". The new prerequisites will be "CSC 135". This update will ensure students can properly enroll in the course as required. Computer Science faculty agree that the new prerequisite is sufficient for the course.

**NEW COURSE DESCRIPTION:** (Not to exceed 80 words and language should conform to catalog copy. See Guidelines for Catalog Course Description [http://www.csus.edu/ummanual/AcadAft/FSC00060.htm](http://www.csus.edu/ummanual/AcadAft/FSC00060.htm))

**Note:**

Prerequisite: CSC 135
Enforced at Registration: Yes _x_ No _x_

Corequisite:
Enforced at Registration: Yes _x_ No _x_

Graded: Letter _x_ Credit/No Credit _x_

Instructor Approval Required? Yes _x_ No _x_

Course Classification (e.g., lecture, lab, seminar, discussion): Title for CMS (not more than 30 characters)

Discussion

Cross Listed?
Yes _x_ 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? _x_ 1

Can the course be taken for Credit more than once during the same term? Yes _x_ 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.”

**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:

For whom is this course being developed?
Majors in the Dept __ Majors of other Dept __ Minors in the Dept __ General Education __ Other __

Is this course required in a degree program (major, minor, graduate degree, certificate)? Yes __ No __
If yes, identify program(s):

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 __
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: ____________________________ Date: 3/6/2015
College Dean or Associate Dean: ____________________________ Date: 2/9/15

University Committee:

Assoc. Dean for Undergraduate Studies OR Dean of Graduate Studies:

Distribution: Academic Affairs (original), Department Chair and College Dean. Dean's office to send original after approval to Academic Affairs, at mail zip 6016.

11/08/2012
COURSE DESCRIPTION

Dept., Number  CSC 151  Course Title  Compiler Construction
Semester hours  3  Course Coordinator  Ted Krovetz
URL (if any):  http://krovetz.net/csus/

Catalog Description

A practical approach to compiler design and implementation. Organization of a compiler, algorithms for lexical, syntactic and semantic analysis, recursive descent and/or LALR parsing, organization of symbol tables, error detection and recovery, object code generation. Modular design will be emphasized. Prerequisites: CSC 135.

Textbook


Course Goals

To develop an understanding of the practical aspects of compiler design and construction, including:

1. At least one parsing technique and its conditions of application.
2. Error recovery.
4. Translation of typical programming language constructs.

Prerequisites by Topic

Thorough understanding of:
- Advanced data structures.
- An object-oriented programming language.

Basic understanding of:
- Regular expressions, finite automata, context-free grammars, pushdown automata.
- Concept of parsing.
- Representation of programming language syntax, BNF, EBNF, syntax diagrams.

Major Topics Covered in the Course

1. Introduction to compilers (1 hour).
2. Source handling (1 hour).
3. Symbol tables (2 hours).
4. Lexical analysis, including the use of finite automata (3 hours).
5. Syntactic analysis, including simulation of push down automata (10 hours).
7. Error recovery (4 hours).
8. Semantic analysis (8 hours).
9. Translation of source to an intermediate language, e.g., Polish postfix, expression tuples (3 hours).
10. Translation of intermediate language to object code (3 hours).
11. Run-time environments (2 hours).
12. Optimization of object code (2 hours).

Outcomes

Thorough understanding of:
- The concepts of scanning and parsing.
- The organization of a compiler.
- Scanning and the use of scanner generators.
- Modular development of a significant programming system.

Basic understanding of:
- Parsing: bottom-up parsing techniques including recursive and LL (1); top-down parsing including SLR (1) and LALR (1).
- Error recovery associated with various parsing techniques.
- Semantic analysis, attribute grammars.
- Method of intermediate code representation.
- Translation of some basic language constructs such as recursion, parameter passing.

Exposure to:
- Code optimization.

Laboratory Projects

Implementation of several portions of a compiler. Such project(s) might include the implementation of:

1. A source handler (2 weeks).
2. A lexical analyzer (minimum 2 weeks).
3. A syntax analyzer (minimum 3 weeks).
4. A semantic analyzer (minimum 2 weeks).
5. A code generator (minimum 3 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>0.5</td>
<td></td>
<td>Data Structures</td>
<td>0.5</td>
<td></td>
</tr>
<tr>
<td>Software Design</td>
<td>0.5</td>
<td></td>
<td>Prog. Languages</td>
<td>1.0</td>
<td></td>
</tr>
<tr>
<td>Comp. Arch.</td>
<td>0.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Oral and Written Communications

No significant component.

Social and Ethical Issues

No significant component.

Theoretical Content

1. Regular expressions and finite automata, automatic generation of lexical analyzers (4 hours).
2. Formal techniques for parsing context-free languages (6 hours).

Problem Analysis

In order to write some portions of a compiler (lexical analyzer, syntax analyzer, semantic analyzer, or code generator), the student must analyze how the techniques discussed in class relate to the particular language to be processed.

Solution Design

Students implement a compiler for a simple programming language using object-oriented techniques and a highly modular design.

/aa