CS 536 — Spring 1999

Course Description and Overview

Instructor:

Charles N. Fischer		
office:	5397 CS Building	
office hours:	Tuesday	1:00-3:00
	Thursday	1:00-3:00
	or	by appointment
phone:	262-6635	
e-mail:	fischer@cs.wisc.edu	

TA:

Jun Shan		
office:	3360 CS Building	
phone:	262-9275	
office hours:	Wednesday	4:00-5:00
	Friday	2:30-4:30
	or	by appointment
e-mail:	jshan@cs.wisc.edu	

Class schedule:

Lecture:

Time: 1:20 - 2:10 MWF

Place: 1227 Engineering

Recitation:

Time: 3:30 - 4:20 Wed

Place: B 317 Chemistry

Course Overview:

CS 536 is an introduction to the design and translation of programming languages. Subjects to be covered include symbol tables, scanning and regular expressions, Lex and Flex, context-free grammars and parsing, Yacc and Bison, abstract syntax trees, code-generation, the Java Virtual Machine, and syntax-directed compilation. A variety of tools, including JLex, JavaCUP and Jasmin will be used. Concepts of name scoping, modularization, and type rules will be considered in a number of languages.

Class projects will be components of a simple compiler for the language CSX, a blend of C, C++ and Java. Projects will be implemented in Java.

Grade Weights (tentative):

45% projects, 45% exams, 10% homework assignments

Textbook:

Crafting a Compiler with C, by C.N. Fischer and R.J. LeBlanc, Benjamin/Cummings, 1991.

A number of handouts, covering assignments and supplementary information, will available at the MACC document sales facility or on the World-Wide-Web .

Supplemental Texts:

(1) *Compilers: Principles, Techniques, and Tools* by A.V. Aho, R. Sethi, and J.D. Ullman, Addison-Wesley, 1986.

(2) *The Java Programming Language, Second Edition* by K. Arnold and J. Gosling, Addison-Wesley, 1998.

Exam Dates:

(1) Midterm: Wednesday, March 17, 1999 (tentative)
(2) Final: Tuesday, May 11, 1999, 12:25PM - 2:25 PM

Programs:

(1) Symbol Table Classes

Due: Wednesday, February 3, 1999

(2) CSX Scanner

Due: Wednesday, February 24, 1999 (tentative)

(3) CSX Parser

Due: Wednesday, March 24, 1999 (tentative)

(4) CSX Type Checker

Due: Wednesday, April 14, 1999 (tentative)

(5) CSX Code Generator

Due: Wednesday, May 5, 1999

Programming assignments will be done in Java. Tutorials covering the essentials of Java will be presented during the first few class meetings. The computers assigned for class projects are Sun workstations running Solaris (named sol1 - sol40 and vega1 - vega36), in rooms 1358, 1363, 1368 and 1370 of the Computer Sciences Building. You may use your own computer if you prefer, but you should login to your departmental account from time to time to read your mail. Note also that we'll be using tools like Jlex, JavaCUP and Jasmin that probably aren't available on personal computers. Tutorials covering Unix will be held during the first few weeks of class.

Programs are due **in class** on the due date. No program will be accepted outside of class. The last program of the semester will not be accepted after the last day of classes. Other programs will be accepted up to one week (three class periods) late. Each class period that a program is late will result in a 10 point late penalty. Each student will be allowed three “free” late penalties per semester. **No program will be accepted more than three class periods late.**

Designated Notetakers:

As a convenience to class members, each class lecture and recitation will be summarized by a volunteer note taker, who will prepare an online file describing the material discussed, no later than one week after the lecture. These summaries will be available to all class members in the directory ˜cs536-1/public/classnotes (and also through the CS 536 homepage). Each class member is be asked to volunteer no more than once. As a reward, each volunteer will be given three additional free late penalties for class projects. To enter a notes file into the classnotes just copy it (using cp). Once the file is entered into the classnotes directory it can’t be changed or removed, so proof your file before you enter it. Also, be sure you’ve allowed read permission to all users. The command

```
chmod a+r yourfile
```

will do this. Name your file to reflect the date of the lecture it covers, using the form mmmdd, where mmm is a three letter month abbreviation (Jan, Feb, Mar, Apr or May) and dd is a two digit date. Thus a file might be named Jan20 or Apr01.

Program Grading:

Programs will be graded according to their *correctness, efficiency, style*, and *readability*, and according to the quality and thoroughness of the test data.

Correctness

Programs should behave as defined in the assignment handouts. Bad data should be handled gracefully; your program should never have runtime errors like dereferencing a null pointer or using an out-of-bounds index. Special cases should be handled correctly.

Efficiency

Unnecessarily inefficient algorithms or constructs should be avoided; however, efficiency should never be pursued at the expense of clarity or simplicity.

Style

- Programs should be well documented. This includes both external and internal documentation. External documentation goes at the head of each module and should be sufficiently informative so that someone could use the module without looking at anything other than this documentation. Internal documentation goes at the beginning of each procedure to explain its behavior, and also goes with any non-obvious piece of code.
- Programs should be modular.
- Programs should be flexible, *i.e.* easy to modify.
- Indentation should reflect program structure.
- Use meaningful identifiers.
- Label end braces (*e.g.* } // while (! done)).

Readability

The grader should be able to understand the program without undue strain. It is not the grader’s responsibility to decipher clumsy or arcane programs.

Test data
Your test data should completely demonstrate the capabilities of your program. Every statement in your program should be exercised by some piece of test data; this includes code included to handle special cases and bad input data. Quality of test data may count for as much as one-third of a program grade. Thorough testing is essential to establish the reliability of a module or class. The time to detect and fix errors is when a program is being developed, *not* when it is in use. The test data we provide or suggest may be incomplete—augment it if necessary.

Extra Credit

The overriding rule about extra credit and fancy features is: don't even think about adding them until the required work is completely debugged. A correctly working simple program is worth much more (both in this class and in actual practice) than a fancy program with bugs. Extra credit will be scored separately from the basic program grade, and will be used subjectively at the end of the term to raise the grades of those who are near the borderline.

Reading Schedule

The following schedule of readings from the textbook (*Crafting a Compiler with C*) is approximate, but it should give you a good idea of what will be covered and when. It is in terms of lecture numbers (there are 43 lectures scheduled for this semester).

Lecture Number	Reading Assignment	Lecture Number	Reading Assignment
1	Ch 1, Ch 2, § 8.1-8.2	4	§ 3.1-3.3, 3.4.2
7	§ 3.5	11	§ 3.6
15	Ch 4	18	§ 5.1-5.3
20	§ 5.4-5.6, 5.8	22	§ 5.7, 5.9
24	§ 8.3, 8.4	27	§ 9.1-9.3
32	§ 10.1, 10.2	38	§ 11.1-11.2, 12.1-12.2, 13.1-13.4

WWW Homepage

A wide variety of information pertaining to this class (handouts, assignments, announcements, readings, old exams, tutorials, etc.) will be available through the CS 536 homepage at `http://www.cs.wisc.edu/~cs536-1/cs536.html`. You may use Netscape, or any comparable browser. You should add the 536 homepage to your list of "bookmarks" as you'll need to reference it frequently throughout the semester.