CS 538  

**Introduction to the Theory and Design of Programming Languages**  

Charles N. Fischer  

Spring 2007  

http://www.cs.wisc.edu/~fischer/cs538.html  

---  

**Class Meets**  
Mondays, Wednesdays & Fridays,  
11:00 — 11:50  
1325 Computer Sciences  

**Instructor**  
Charles N. Fischer  
6367 Computer Sciences  
Telephone: 262-6635  
E-mail: fischer@cs.wisc.edu  
Office Hours:  
10:30 - Noon, Tuesdays & Thursdays, or by appointment  

---  

**Teaching Assistant**  
Mark Goadrich  
1302 Computer Sciences  
Telephone: 262-6600  
E-mail: richm@cs.wisc.edu  
Office Hours:  
1:30 - 3:00, Wednesdays and Fridays, or by appointment  

---  

**Key Dates**  
- Feb 21: Homework #1 (tentative)  
- March 19: Programming Assignment #1 - Scheme (tentative)  
- March 28: Midterm Exam (tentative)  
- April 13: Programming Assignment #2 - Standard ML (tentative)  
- April 30: Programming Assignment #3 - Prolog (tentative)  
- May 11: Programming Assignment #4 - Java, C#, Pizza and Python  
- May 17: Final Exam 2:45pm-4:45pm
Class Text

- Required text:

- Handouts and Web-based reading will also be used.

Reading Assignment

- Webber: Chapters 1, 10, 18 (as background)

Class Notes

- Each lecture will be made available prior to that lecture on the class Web page (under the “Lecture Nodes” link).

Instructional Computers

Departmental Linux Machines (royal1-royal30, emperor01-emperor40) have been assigned to CS 538. All necessary compiler, interpreters and tools will be loaded onto these machines.

You may also use your own PC or laptop. It will be your responsibility to load needed software (instructions on where to find needed software are included on the class web page).

The Systems Lab teaches brief tutorials on Linux if you are unfamiliar with that OS.

Academic Misconduct Policy

- You must do your own assignments — no copying or sharing of solutions.
- You may discuss general concepts and ideas.
- All cases of misconduct must be reported to the Dean’s office.
- Penalties may be severe.
Program & Homework Late Policy

- An assignment may be handed in up to 7 days late, but no later.
- Each day late will be debited 4%, up to a maximum of 28%.
- All students are given 10 “free” late days. That is, the first 40% in late debits will be automatically forgiven.
- Your 10 free late days may be used at any time, and in any combination.

Approximate Grade Weights

<table>
<thead>
<tr>
<th>Component</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Homework 1</td>
<td>10%</td>
</tr>
<tr>
<td>Program 1 - Scheme</td>
<td>16%</td>
</tr>
<tr>
<td>Program 2 - ML</td>
<td>16%</td>
</tr>
<tr>
<td>Program 3 - Prolog</td>
<td>12%</td>
</tr>
<tr>
<td>Program 4 - Java, C#, Pizza, Python (optional extra credit)</td>
<td>10%</td>
</tr>
<tr>
<td>Midterm Exam</td>
<td>23%</td>
</tr>
<tr>
<td>Final Exam (non-cumulative)</td>
<td>23%</td>
</tr>
</tbody>
</table>

Programming Languages to be Considered in Detail

1. Scheme
   A modern variant of Lisp.
   A *Functional Language*: Functions are “first class” data values.
   *Dynamically Typed*: A variable’s type may change during execution; no type declarations are needed.
   All memory allocation and deallocation is *automatic*.
   Primary data structures, lists and numbers, are *unlimited* in size and may grow without bound.

2. ML ("Meta Language")
   Strong, compile-time type checking.
   Types are determined by *inference* rather than declaration.
   Naturally polymorphic (one function declaration can be used with many different types).
   Pattern-directed programming (you define patterns that are automatically matched during a call).

Continuations provide a novel way to suspend and “re-execute” computations.
Typed exceptions are provided. Abstract data types, with constructors, are included.

3. Prolog (Programming in Logic)
Programs are Facts and Rules. Programmers are concerned with definition, not execution. Execution order is automatically determined.

4. Pizza
Extends a popular Object-oriented language, Java, to include
- Parametric polymorphism (similar to C++'s templates).
- First-class functional objects.
- Algebraic data types, including patterns.

5. C#
Microsoft’s answer to Java. In most ways it is very similar to Java, with some C++ concepts reintroduced and some useful additions.

- Events and delegates are included to handle asynchronous actions (like keyboard or mouse actions).
- Properties allow user-defined read and write actions for fields.
- Indexers allow objects other than arrays to be indexed.
- Collection classes may be directly enumerated: ```foreach (int i in array) ...``` 
- Structs and classes co-exist and may be inter-converted (boxed and unboxed).
- Enumerations, operator overloading and rectangular arrays are provided.
- Reference, out and variable-length parameter lists are allowed.

6. Java 1.5 (Tiger Java, Java 5.0)
Extends current definition of Java to include:
- Parametric polymorphism (collection types may be parameterized).
- Enhanced loop iterators.
- Automatic boxing and unboxing of wrapper classes.
- Typesafe enumerations.
- Static imports (`out.println` rather than `System.out.println`).
- Variable argument methods.
- Formatted output using `printf`: ```out.printf("Ans = %3d", a+b);```
7. Python
A simple, efficient scripting language that quickly builds new programs out of existing applications and libraries. It cleanly includes objects. It scales nicely into larger applications.

Evolution of Programming Languages

In the beginning, ... programs were written in absolute machine code—a sequence of bits that encode machine instructions. Example:

```
34020005
0000000c
3c011001
ac220000
```

This form of programming is
- Very detailed
- Very tedious
- Very error-prone
- Very machine specific

Symbolic Assemblers
Allow use of symbols for operation codes and labels.
Example:

```
li $v0,5
syscall
sw $v0,a
```

Far more readable, but still very detailed, tedious and machine-specific.
Types are machine types.
Control structures are conditional branches.
Subprograms are blocks of code called via a “subroutine branch” instruction.
All labels are global.