Syllabus
Winter 2009

Course: CS180h: Scheme
Time: TTh 3rd hour
Room: Cat Lab SMC E-011
Website: http://courses.knox.edu/cs180h/

Professor: Don Blaheta
Office: SMC E-210
Phone: x7956
Email: dblaheta@knox.edu
Office hours: TBA

General info

The Scheme language is a canonical modern example of a functional pro-
gramming language; along with other dialects of LISP, it is in active use
in artificial intelligence and programming language research, and is a com-
ponent in a variety of modern computer applications. In addition to the
basics of Scheme semantics, we will discuss the relevance of immutable ob-
jects, functions as first class values, lexical closures, and the use of macros
to create new language constructs on the fly.

Book and environment

The book for this course will be Kent Dybvig’s The Scheme programming
language. It is a relatively terse introduction to the language, plus a conve-
nient reference for the basic language library.

The programming environment we’ll use is DrScheme. It is installed on the
lab machines, but if you wish to work from home, you should be able to
download it (v4.1.x) from www.drscheme.org.
Meeting days

This course is fairly tightly interleaved with the AI course, which also meets 3rd hour. Although this course is billed as Tuesday-Thursday, after the first few weeks, the AI course takes over the Thursday time slot for its labs. My current plan is that this course will meet Tuesday-Thursday all through January, and then Tuesdays only for the first three weeks of February; I'll let you know if that changes. The last few weeks of the term we won’t meet as a group, but you’ll be able to use that time to work on the second project.

Graded work

This course is all about the programming, and you’ll be evaluated entirely on programs you write.

Homework. I’m planning on giving a homework assignment every class, due at the beginning of the following class. That’s because there’s really no way to learn this stuff except by doing it. Most of the problems will require just a few lines of code, with perhaps one helper function, but there will be many of these in each assignment. It’s important that you keep up with this. Generally, homework grading will be more-or-less “black box”—that is, given various inputs, does your code behave correctly—but you should also be following our coding conventions and practicing good design and style.

Projects. There will be two larger projects as well. These will require a significant design component, and you’ll have about a week and a half for each one. In addition to correct behaviour, your code will need to be (relatively) efficient, well-laid-out, and conform to the style guidelines.

<table>
<thead>
<tr>
<th></th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Homework</td>
<td>50%</td>
</tr>
<tr>
<td>Projects</td>
<td>50% (25 each)</td>
</tr>
</tbody>
</table>
Topics (tentative)

Week 1
Prefix notation, define, calling and defining functions, test cases in DrScheme; Symbols and quote, Scheme’s evaluation paradigm, basic data types, Conditional execution with cond, Lists

Week 2
List recursion; Function abstraction

Week 3
Vectors, accumulation and tail recursion Generative recursion;

Week 4
Mutation; Closures

Week 5
Macros

Week 6
Continuations

Week 7
The lambda calculus

Policies

Collaboration policy
All the assignments in this course are what I call “lightly collaborative”—for a description of what this means, and some examples of acceptable and
unacceptable behaviour, please go to http://faculty.knox.edu/dblaheta/collab.html. In short, I encourage you to discuss the problems amongst yourselves, but you have to write out the answers each on your own.

Read the spec!

Because of the way I’ll be grading the homeworks, it will be important to carefully follow the specifications given in a problem, calling your functions by the correct name, taking the correct inputs, and so on.

If I’ve got time and I’m feeling generous, I may fix your spec violations so that I can grade the rest of your work. In such cases, you’ll most likely get a 10% penalty, even (perhaps especially) in code that otherwise works very well. If I’m in a hurry or being grumpy, your spec-violating code will be graded on how many test cases they pass, i.e. none of them, i.e. you get a zero on that problem. So read the spec.

Attendance and late policy

Attendance is required, and assignments must be turned in on time. That said, if you have a good reason to miss class or hand something in late, I tend to be fairly easygoing about it; but I need to hear about it in advance. (Medical and family emergencies excepted, of course. But get a note from a dean.) Extensions of a day or two will be granted fairly routinely, as long as the reason is something better than “I didn’t start until today, and now I can’t finish on time!” If in doubt, come talk to me. Frequent absence will result in a lowered participation grade; habitual absence may in extreme cases result in a failing grade for the class. Unexcused late assignments will normally be given a zero.

Early bird policy

My standard early bird policy applies: if you start early and find an error in the problem definition (for homework, project, exam, whatever), and track it down and report it, you’ll get extra credit of some variety. See http://faculty.knox.edu/dblaheta/policies.html#early for details.