COMP 210, FALL 2000
Lecture 1: Administrivia & Introduction

This course is titled "Principles of Computing and Programming."

The Web Site

http://www.owlnet.rice.edu/~comp210

Instructors: Keith D. Cooper, 2065 DH, keith@rice.edu
John Greiner, 3118 DH, greiner@cs.rice.edu

Teaching Assistants: Tim Harvey, 2064 DH, harv@cs.rice.edu
Todd Waterman, 2069 DH, waterman@cs.rice.edu

The five undergraduate laboratory assistants have experience grading this course.

The Book:

Draft of a new textbook (MIT PRESS, 2000 or 2001)
How to Design Programs, by Felleisen, Flatt, Findler, & Krishnamurthi
Available at the bookstore
Older draft is online; you want the newer draft, with corrections.

Requirements:

1. Attend class
2. Attend lab section
   - Wednesday afternoon & Thursday afternoon
   - Taught by John, Tim, Todd, 2 senior labbies
   - Homework returned in lab section
   - Covers some topics skipped in class, reinforces others
   - First labs are next week, sign up quickly (on the web site)
3. Weekly homework
   - Handed out Wednesday due following Wednesday (start of class)
   - Some programming
   - Some paper exercises
   - Done in teams (more later on this subject)
   - Lab section should help prepare you for homework
   - Homework should prepare you for the tests
4. Final Project (a larger program)
5. Three exams
   - Roughly five week intervals
   - First one is in-class, others may be at night

Basis for Grading:

40% from homework
10% from final project
10% from first exam
20% from second exam
20% from third exam

Homework Teams

We encourage you to work with a partner on your homework. You pick the partner; if you're having trouble finding one, we'll find you one. Rules are simple: work on all aspects of the homework—ideally, solve them all yourself. Compare them with your partner. Hand in the best solution.

Hand in exactly one solution, with both your names on it. You should not be in two different teams. You should not hand in two copies of the same homework.

If you **NEED** to work alone, come talk to either John or me.

Programming

All programming will be done in Scheme, using the Dr. Scheme programming environment. It is installed on OwlNet; when you run the registration software, it will set up your UNIX PATH so that Dr. Scheme is on it.

If you want a copy of Dr. Scheme for your personal computer (or your work computer), you can retrieve it from the World-Wide Web. Follow the pointers on the COMP 210 web page. It runs on most common computers.

Keeping Up with Class

Check the web site regularly.
Check the newsgroup. Post your questions, comments, and musings.
Attend class and attend your lab section.
Read the text **BEFORE** we cover it in class.

ACTION ITEMS for Students

Get an OwlNet account
Visit the course web site and register for a lab section
Start thinking about a partner for the homeworks

Now, for some CONTENT (or, at least, some motivation)

What is a computation?

When we speak of computation, we usually mean the automatic evaluation of some algebraic expression.

For example,, if a pizza costs $12 and has 8 slices, what is your share of the bill for the pizza?

We need to know how many slices you ate

- 1 slice implies you owe (12 / 8) * 1 = $ 12/8 or $1.50
- 2 slices implies you owe (12 / 8) * 2 = $ 24/8 or $3
- 12 slices implies you owe (12 / 8) * 13 = $144/8 or $18

To pay for your pizaa, you might be considering a work-study job that pays $5.65 per hour. What would be your gross pay?

We need to know the number of hours worked

- 2 hours implies 5.65 * 2 = 11.30
- 10 hours implies 5.65 *10 = 56.50
- 12 hours implies 5.65 * 12 = 67.80

In a completely different vein, what if someone asks you for the surface area of the pizza? We need to know its radius

- radius of 1 cm implies pi * (1 * 1) = pi * 1 = ~ 3.14…
- radius of 10 cm implies pi * (10 * 10) = pi * 100 = ~ 314…
- radius of 11 cm implies pi * (11 * 11) = pi * 121 = ~ 380 (380.1327….)

To make a computer perform these calculations, we need to write down a rule that it can follow. For our examples

If you worked H hours, your gross pay is 5.65 * H

If you at S slices of pizza, you owe (12 / 8) * S.

If the radius of a disk is R, its area is pi * (R * R) [or R^2]

The languages that we use to express these rules to a computer have stricter notions of syntax than that—mostly to allow the computer to parse the rules automatically. These languages are called programming languages. In COMP 210, we will use a programming language with a particularly simple syntax, Scheme.

Example computations in Scheme:

Rule	Scheme Form	
5	5	
1.20 * 2	(* 1.20 2)	[Prefix notation]
6 * pi * (3 * 3)	(* 6 (* pi (* 3 3)))	
5.65 * 10	(* 5.65 10)	
(12/8) * 3	(* (/ 12 8) 3)	
pi * (10 * 10)	(* pi (* 10 10))	

Every non-trivial expression starts with a "(" and a symbol that explains what comes next. Thus, * means a two-operand multiplication, so it should be followed by two expressions. Trivial expressions, like numbers and names don't need parentheses.

These computations, expressed in a programming language such as Scheme, are programs. That is, they are concise, formal specifications that allow the computer and its various layers of software to evaluate your original computation. ARGUMENTS like H, S, and R create parameterized expressions or parameterized programs, so that we can reuse the basic computation with different sets of values.

Computing is the process of evaluating expressions. Computational science generalizes beyond high-school algebra to include algebras that operate over more complex domains, including numbers, names, symbols, and myriad other abstract spaces—(*i.e.*, the space of LR(1) grammars, or the space regular languages, or URLs on the WWW.)

[Note: the * operator in Scheme can actually take an arbitrary number of arguments, as can other operators. We will behave as if it is a binary (two-argument) operator so that it relates more intuitively to your experience with simple algebra.]