CS3:
Introduction to Symbolic Programming

Lecture 13:
Introduction to the big project
Lists

Spring 2006

Nate Titterton
nate@berkeley.edu

Schedule

13	Apr 10-14	Lecture: MIDTERM #2 Lab: Start on "Lists"
14	Apr 17-21	Lecture: The big project Lab: Lists; start on the project
15	Apr 24-28	Lecture: Lists, Lab: Work on the project
16	May 1-5	Lecture: Final Review Lab: Project Due
17	May 9-14	Lecture: Open review session Lab: *NONE (the semester is over)*
18	May	Final: Wednesday, May 18th

Midterm #2

Any questions?

The Big Project

- **Three possible projects:**
  - **Database**
  - **Yukon**
  - **Blocks World**
- **You can, and should, work in partnerships**
- **You will have three weeks to work on this (it is due on the last lab)**
- **Worth 15% of you final grade**

Project Check-offs

- **There are 3 checkoffs**

  You need to do them on time in order to get credit for the project

3. **Tell your TA which project you will do and who you will do it with**
4. **Show your TA that you have accomplished something. S/he will comment.**
5. **Show that you have most of the work done: your TA will run your code.**

Due dates on the final project

Tues/Wed	Thur/Fri
(Apr 18-19) Introduction	*(Apr 20-21)* Checkoff 1
(Apr 25-26)	*(Apr 27-28)* Checkoff 2
(May 2-3) Checkoff 3	*(May 4-5)* Due (at midnight)

Only two more lectures (after this one)…

What would you like to do?

- **Hear about the CS major, and other courses…**
- **Do exam-type problems…**
- **Review…**

On May 9th, I plan on holding an open review session. Other suggestions are welcome!

Lets see the projects in action

What issues matter

- **Does it work?**
  - **This is a primary grading standard…**

- **Programming style**
- **Reading specifications carefully**
- **Error checking inputs (especially in the database project)**
- **Adequate testing**
- **Code reuse (again, with the database)**

Working in partnerships

- **Highly recommended!**
  - For those of you continuing with CS, you'll be doing this for many future projects
- **Won't be faster, necessarily**
  - While you are less likely to get stuck, there will be a lot of communication necessary
- **A big benefit will be with testing**
- **Remember, only one grade is given…**
  - this grade will be the same, whether the project is a solo or a partnership

Functional Programming

- **In CS3, we have focused on programming without *side-effects*.**
  - **All that can matter with a procedure is what it returns**
  - **In other languages, you typically:**
    - **Perform several actions in a sequence**
    - **Set the value of a variable – and it stays that way**
  - **All of this is possible in Scheme.**

Printing, and sequencing

- **With *Blocks World* and *Yukon* you will need to display information.**
  - ***Simply Scheme* chapter 20 is nice summary.**
  - **And, all the projects have file input /output routines that you don't need to "understand", as well as user input routines.**

Data structures

- **The format of data used in these projects in a central feature**
  - **A "data structure" (abstract data type) is a specification of that format. Here, generally, lists of lists (of lists).**
  - **Accessors and constructor allow for *modularity*: letting parts of a program work independently from other parts.**

Strings versus words

- **One useful data structure is a string**
  - Strings are surrounded by double quotes when printed.
  - Strings are a native type in Scheme.
- **In CS3, you used words (sometimes sentences) to present names and other output to the user.**
  - In the real world, strings are used.

Lists

- **Lists are containers, like sentences, where each element can be anything**
  - **Including, another list**

```
((beatles 4) (beck 1) ((everly brothers) 2) … )

((california 55) (florida  23) ((new york) 45) )

(#f   #t   #t   #f   #f  …)
```

Sentences(words) vs lists: constructors

cons Takes an element and a list Returns a list with the element at the front, and the list contents trailing	
append Takes two lists Returns a list with the element of each list put together	
list Takes any number of elements Returns the list with those elements	sentence Takes a bunch of words and sentences and puts "them" in order in a new sentence.

Sentences(words) vs lists: selectors

car Returns the first element of the list	first Returns the first word (although, works on non-words)
cdr Returns a list of everything but the first element of the list	butfirst Returns a sentence of everything but the first word (but, works on lists)
	last …
	butlast …

Sentences(words) vs lists: HOF

map Returns a list where a func is applied to every element of the input list. Can take multiple input lists.	every Returns a sentence where a func is applied to every element of an input sentence or word.
filter Returns a list where every element satisfies a predicate. Takes a single list as input	keep Returns a sentence or word where every element satisfies a predicate
reduce Returns the value of applying a function to successive pairs of the (single) input list	Accumulate Returns the value of applying a function to successive pairs of the input sentence or word