
CS3:

Introduction to Symbolic Programming

Lecture 14: Lists

Scheme vs. other programming languages

Spring 2006

Nate Titterton
nate@berkeley.edu

Schedule

15	Apr 24-28	Lecture: Lists, other languages Lab: Big Project CHECKOFF #2 – Thur/Fri
16	May 1-5	Lecture: Guest Lecture: what is CS at UCB? Lab: Finish up the Project CHECKOFF #3 – Tue/Wed Project Due on Fri (at midnight)
17	May 9-14	Lecture: Review of CS3, solving problems Lab: <i>NONE (the semester is over)</i>
18	May 18	Final: Thursday, May 18 th 12:30 – 3:30, 4 LeConte

Lists

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

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

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

A few other important topics re: lists

2. `map` can take multiple arguments

4. Association lists

6. Generalized lists

map can take multiple list arguments

```
(map + '(1 2 3) '(100 200 300))  
→ (101 202 303)
```

The argument lists have to be the same length

```
(define (palindrome? lst)  
  (all-true?  
    (map equal? lst (reverse lst))))
```

```
(palindrome? '(a m a n a p l a n a c a n a l p a n a m a))  
→ #t
```

Quiz: Can you write `all-true?` without `if` and `cond`?

Generalized lists

- **Elements of a list can be anything, including any list**
- **Lab materials discuss**
 - `flatten (3 ways)`
 - `completely-reverse`
 - `processing a tree-structured directory`

Scheme versus other languages

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; Chapter 20 is a good place to start

The language Scheme

- **Scheme allows you to ignore tedium and focus on core concepts**
 - The core concepts are what we are teaching!
- **Other languages:**
 - Generally imperative, sequential
 - Lots and lots of syntactic structure (built in commands)
 - Object-oriented is very "popular" now

CS3 concepts out in the world

- **Scheme/lisp does show up: scripting languages inside applications (emacs, autocad, Flash, etc.)**
- **Scheme/Lisp is used as a "prototyping" language**
 - **to quickly create working solutions for brainstorming, testing, to fine tune in other languages, etc.**
- **Recursion isn't used directly (often), but recursive ideas show up everywhere**

Java and PHP

- **Java is a very popular programming language**
 - Designed for LARGE programs
 - Very nice tools for development
 - Gobs of libraries (previous solutions) to help solve problems that you might want solved
- **PHP**
 - Popular course for web development (combined with a web-server and database)
 - Lots of features, but little overall "sense"
 - Because programs in PHP execute behind a web-server and create, on the fly, programs in other languages, debugging can be onerous.

SQL resembles HOFs

- SQL if for database retrieval
- query: “Tell me the names of all the lecturers who have been at UCB longer than I have.”

```
select name from lecturers
  where date_of_hire <
    (select date_of_hire from lecturers where name =
      'titteton');
```

- query: “Tell me the names of all the faculty who are older than the faculty member who has been here the longest.”

```
select L1.name from lecturers as L1 where
L1.age >
  (select L2.age from lecturers as L2
   where L2.date_of_hire =
     (select min(date_of_hire) from lecturers) );
```

Problems

A list version of electoral-votes

Write a higher-order procedure named `electoral-votes` which takes a predicate as its single argument. The procedure will sum up the 2008 electoral votes for states that satisfy the predicate.

```
(electoral-votes california?) ➔ 55  
(electoral-votes blue-state?) ➔ 212
```

The database of states and their electoral votes is in a global variable `*states*`:

```
((ca 55) (me 4) (nj 15) ...)
```

The predicate takes the state's two-letter abbreviated name as its argument. You do not have to write these predicates; rather, you only need to write `electoral-votes` such that it works properly with any proper predicate.