CS151.01 2009F *Functional Problem Solving*

Class 42: Association Lists

Held: Monday, 16 November 2009

Summary: We consider *association lists*, a simple, but useful, technique for organizing tables of information.

Related Pages:

- EBoard.
- Lab: Association Lists.
- Reading: Association Lists.

Notes:

- Reading for Tuesday: Design Patterns and Higher-Order Procedures. (Warning: You may find that it repeats some things you learned earlier in this semester. That's okay, we want you to have the reinforcement.)
- Are there questions on the project?
- Fridays *S&B* article on the campus climate survey reinforced to me that I want to spend a bit more class time on that survey. (Executive summary available from `http://www.grinnell.edu/offices/president/diversity/`.)

Overview:

- Storing information in tables.
- Representing table entries as lists.
- Representing tables as lists.
- Association lists: Scheme's standard table representation.
- Implementing key association list procedures.

Simple Database Problems

- Databases are among the most common applications of computers.
  - After all, there's a reason that Larry Ellison is nearly as rich as Bill Gates.
- A database is a mechanism for storing data so that you can easily access the data you need.
- One simple database activity is looking up values by *keys*.
- Databases that provide only that activity are called *dictionaries*.

Association Lists

- In Scheme, dictionaries are typically implemented with a data structure known as the *association list*.
- An association list is a list of elements each of which has a key as its car.
- You can use the `(assoc key list)` procedure to look up values by key.

Searching in Association Lists

- Suppose Scheme didn't include `assoc`. How would you write it? Probably *recursively*.
- If the list is empty, it does not contain the value.
- If the key of the first element in the list is the key we're looking for, return the corresponding value.
- Otherwise, look in the rest of the list.
- This technique is called *sequential search*.

Variants of Association Lists

- Given that we can write our own `assoc` procedure, we can easily implement a number of interesting variants of association lists.
- For example, if the same key appears multiple times in the association list, we might return *all* matching values (rather than the *first* matching value).
- Similarly, instead of searching by key, we might search by predicate.

alists-lab

- Do the lab.
