COMP 200: Elements of Computer Science
Fall 2004
Homework 3 Due Monday, September 27, 2004

Follow the homework guidelines posted on the website

This homework requires you to program using Dr. Scheme. You can find Dr. Scheme on the Owlnet PC and MacIntosh machines. (If you have trouble, contact the laboratory assistants.) Alternatively, you can download Dr. Scheme for your home machine from http://www.drscheme.org.

To see a clear example of how to hand in the assignment, look at the solution to Homework 2 (on the web page).

Using Scheme's Builtin List Facility

In class, we developed a data definition for *list-of-numbers.* In truth, lists are an important feature of Scheme. Scheme provides a builtin definition for a list data type. The builtin list resembles our list of numbers, except that:

- It has different names for the fields and selectors.

  ```
  ; a list is either
  ;    empty
  ; or a structure
  ;   (cons first rest)
  ; where first is any Scheme object and rest is a list
  ;;; it is builtin, so we do not need a "define-struct" to create it.
  ```

  The selectors are simply **first** and **rest**. The constructor is simply **cons**.

- The first element of a list may be of any type, where we restricted that field to numbers.

You may use the builtin list constructor rather than defining your own version. For example, the **Total** program from Lecture 10 would be written

```
; Total : list of numbers → number
; Purpose: sum the elements of an input list
(define (Total fee)
   (cond
     ((empty? fee)   0)
     (else
          (+ (first fee) (Total (rest fee))))
   ))
```

Programming with Lists

1. Develop a program ***List-search*** that takes as input a list of numbers and a single number and returns the value true if the key (its single-number input) occurs at least once in the input list and the value false if the key does not occur in the list.

```
; List-search : list-of-numbers   number → true or false
; Purpose: returns true if the solo number (key) occurs in the list-of-numbers
;              and false otherwise
(define  (List-search alist key) … )
```

Be sure to show all the steps in the methodology, including contract, purpose and header, test data, your final code, and a transcript of the interactions window showing the results on your test data.

2. Develop a program ***List-large*** that takes as input a list of numbers and returns a count of all the list elements that are greater than or equal to 100 in value.

```
; List-large : list-of-numbers  number → number
; Purpose: counts the number of elements whose value is greater than
;              or equal to 100
(define  (List-large  alist) … )
```

Be sure to show all the steps in the methodology, including contract, purpose and header, test data, your final code, and a transcript of the interactions window showing the results on your test data.