MASSACHVSETTS INSTITVTE OF TECHNOLOGY
Department of Electrical Engineering and Computer Science
6.090—Building Programming Experience
IAP 2005

Lecture 4

Scheme

1. **Special Forms**

   (a) *define* (sugared form) - `(define (`*name parameters*`)` *expressions*`)`
   This form is equivalent to `(define` *name* `(lambda (`*parameters*`)` *expressions*`))`.

   (b) *let* - `(let` *bindings body*`)`
   Binds the given bindings for the duration of the body. The bindings is a list of (*name value*) pairs. The body consists of one or more expressions which are evaluated in order and the value of last is returned.

Problems

1. Guess the value, then evaluate the expression in scheme. If your guess differs from the actual output, try desugaring any relevant expressions.

```
(define (foo x)
  (+ x 3))

foo

(foo 5)

(define bar 5)

(define (baz) 5)

bar

baz
```

```
(bar)

(baz)

(let ((a 3)
      (b 5))
  (+ a b))

(let ((+ *)
      (* +))
  (+ 3 (* 4 5)))

(define m 3)
(let ((m (+ m 1)))
  (+ m 1))

(define n 4)
(let ((n 12)
      (o (+ n 2)))
  (* n o))
```

Data Structures

New procedures

1. `(cons a b)` - Makes a cons-cell (pair) from a and b
2. `(car c)` - extracts the value of the first part of the pair
3. `(cdr c)` - extracts the value of the second part of the pair
4. $(\texttt{c}\frac{a}{d}\frac{a}{d}\frac{a}{d}\frac{a}{d}\texttt{r c})$ - shortcuts
5. `(list a b c ...)` - builds a list of the arguments to the procedure
6. `(list-ref lst n)` - returns the nth element of lst
7. `(append l1 l2)` - makes a new list containing the elements of both lists
8. `(null? lst)` - is `lst` the empty list?

Problems

2. Draw box-and-pointer for the values of the following expressions.

```
(list 1 2 3)
```

```
(cons 3 (list 1 2))
```

```
(cons 1 (cons 3 (cons 5 nil)))
```

```
(list (list 3))
```

3. Write expressions whose values will print out like the following.

   ```
   (1 2 3)
   ```

   ```
   ((1 2) (3 4) (5 6))
   ```

   ```
   ((4 7) 2)
   ```

4. Write expressions using `car` and `cdr` that will return 4 when the `lst` is bound to the following values:

   ```
   (7 6 5 4 3 2 1)
   ```

   ```
   ((7) (6 5 4) (3 2) 1)
   ```

   ```
   (7 (6 (5 (4 (3 (2 (1)))))))
   ```

   ```
   (7 ((6 5 ((4)) 3) 2) 1)
   ```