CS61A – Lab Assignment 2.2 | **Kurt Meinz**
University of California, Berkeley | Summer 2005

1. Try these in Scheme:

```
(define x (cons 4 5))
(car x)
(cdr x)
```

```
(define y (cons 'hello 'goodbye))
(define z (cons x y))
(car (cdr z))
(cdr (cdr z))
```

2. Predict the result of each of these before you try it:

```
(cdr (car z))

(car (cons 8 3))

(car z)

(car 3)
```

3. Enter these definitions into Scheme:

```
(define (make-rational num den)
  (cons num den))

(define (numerator rat)
  (car rat))

(define (denominator rat)
  (cdr rat))

(define (*rat a b)
  (make-rational (* (numerator a) (numerator b))
                 (* (denominator a) (denominator b))))

(define (print-rat rat)
  (word (numerator rat) '/ (denominator rat)))
```

4. Try this:

```
(print-rat (make-rational 2 3))

(print-rat (*rat (make-rational 2 3) (make-rational 1 4)))
```

5. Define a procedure `+rat` to add two rational numbers, in the same style as `*rat` above.

6. Suppose the constructor for rational numbers was changed to

```
(define (make-rational num den)
  (sentence num den))
```

Rewrite the rest of the functions in exercise 3 such that it preserves the behavior of exercises 4 and 5.

7. *SICP* ex. 2.4

8. *SICP* ex. 2.18; this should take some thought, and you should make sure you get it right, but don't get stuck on it for the whole hour. **Note:** Your solution should reverse *lists*, not sentences! That is, you should be using `cons`, `list`, and `append`, not `first`, `butfirst`, `sentence`, etc.