CS61A – Lab Assignment 3.1	**Kurt Meinz**
University of California, Berkeley	Summer 2002

1. *SICP* ex. 2.25 and 2.53; these should be quick and easy.

2. *SICP* ex. 2.55; **explain your answer to your TA.**

3. *SICP* ex. 2.27. This is the central exciting adventure of today's lab! Think hard about it.

4. Each person individually make up a procedure named `mystery` that, given two lists as arguments, returns the result of applying *exactly two* of `cons`, `append`, or `list` to `mystery`'s arguments, using no quoted values or other procedure calls. Here are some examples of what is and is not fair game:

`okay`

```
(define (mystery L1 L2)
  (cons L1 (append L2 L1)))
```

```
(define (mystery L1 L2)
  (list L1 (list L1 L1)))
```

```
(define (mystery L1 L2)
  (append (cons L2 L2) L1))
```

`not okay`

```
(define (mystery L1 L2)
  (cons L1 (cons L2 (cons L1 L2))))
```

```
(define (mystery L1 L2)
  (cons L1 L2))
```

```
(define (mystery L1 L2)
  (append L1 (cons L1 '(A B C))))
```

Type your `mystery` definition into a file, and have one of your partners load it into Scheme and try to guess what it is by trying it out with various arguments.

After everyone has tried someone else's procedure, decide with your partners which procedure was hardest to guess and why, and what test cases were most and least helpful in revealing the definitions.