

**CS 101 Computer Science I (Spring 2001)**  
**Lab 2: January 31 – February 2, 2001**

1. The following Scheme procedure takes one argument and returns a list with two copies of the argument:

```
(define copy (lambda (x) (cons x (cons x '()))))
```

Type this definition into DrScheme. Then have DrScheme evaluate the following expressions and write down the result:

```
(copy 'fred) _____  
(copy (copy 'ethel)) _____  
(copy (copy (copy 'lucy))) _____
```

2. Write a Scheme procedure called "flip" that takes a list of length two as an argument and returns another list with the same two members, but in the reverse order. For example:

```
(flip '(a b))      ==>      (b a)
```

Test your flip procedure by evaluating the following expressions:

```
(flip '(1 2))      (flip '((a)(b)))      (flip (flip '((a b)(c d))))
```

3. Write a Scheme predicate called "num-or-boolean?" that returns #t if its argument is either a number or a boolean, and returns #f otherwise. You must write em two different versions of this procedure --- one version that uses cond or if (but does not use any of and, or & not), and one that uses neither cond nor if (but may use any of and, or & not).

```
(num-or-boolean? 7) ==> #t  
(num-or-boolean? #f) ==> #t  
(num-or-boolean? 'foo) ==> #f
```

4. Write a Scheme predicate called "same-sign?" that takes two numbers as inputs. The procedure returns #t if the two numbers are both positive, both negative, or both zero. It returns #f otherwise. Your procedure only needs to work for numeric arguments. You must write two different versions of this procedure --- one version that uses cond or if (but does not use any of and, or & not), and one that uses neither cond nor if (but may use any of and, or & not).

```
(same-sign? -1 7) ==> #f  
(same-sign? -1 -7) ==> #t  
(same-sign? 0 -7) ==> #f  
(same-sign? 0 0) ==> #t  
(same-sign? 6 7) ==> #t
```