CS61A, Fall 1994
Midterm #3

Question 1 (3 points):

What will the Scheme interpreter print in response to each of the following expressions? Also, draw a ``box and pointer'' diagram for the result of each expression. Hint: It'll be a lot easier if you draw the box and pointer diagram *first*!

```
(let ((x (list 1 2 3 4)))
  (set-car! (cdr x) (cddr x))
  x)

(let ((x (list 1 2 3 4)))
  (set-cdr! (cdr x) (caddr x))
  x)

(let ((x (list 1 '(2 3) 4)))
  (set-car! x (caddr x))
  x)
```

Question 2 (4 points):

Write `make-alist!`, a procedure that takes as its argument a list of alternating keys and values, like this:

```
(color orange zip 94720 name wakko)
```

and changes it, by mutation, into an association list, like this:

```
((color . orange) (zip . 94720) (name . wakko))
```

You may assume that the argument list has an even number of elements. The result of your procedure requires exactly as many pairs as the argument, so you will work by rearranging the pairs in the argument itself. **Do not allocate any new pairs in your solution!**

Question 3 (4 points):

Define a stream named `all-integers` that includes all the integers: positive, negative, and zero.

You may use any stream or procedure defined in the text.

Question 4 (4 points):

We are going to modify the adventure game project by inventing a new kind of place, called a *hyperspace*. Hyperspaces are just like other places, connected to neighboring non-hyperspace places in specific directions, except that they behave strangely when someone enters one: The person who entered is sometimes magically transported to another hyperspace. (The hyperspaces must all know about each other, but they are not connected to each other through exits. Each hyperspace is connected to specific neighbors, just as any place is.)

Your job is to define the `hyperspace` class. The class must be defined in a way that allows you to know all of its instances. When a person enters a hyperspace, half the time nothing special should happen, but half the time the hyperspace should ask the person to `go-directly-to` some randomly chosen other hyperspace.

You may use the following auxiliary procedures if you wish:

```
(define (coin-heads?) (= (random 2) 1))

(define (choose-randomly stuff)
  (nth (random (length stuff)) stuff))
```

Do not modify any existing class definitions.

Group Question (4 points):

Draw the environment diagram for the situation after the following definition and invocation have been evaluated:

```
(define (every-nth num lst)
  (define (help lst pos)
    (cond ((null? lst) '())
          ((= (remainder pos num) 0)
           (cons (car lst) (help (cdr lst) (1+ pos))))
          (else (help (cdr lst) (1+ pos)))))
  (help lst 1))

(every-nth 2 '(she loves you))
```

Take a peek at the solutions