CISC 280 Spring 2007 Lab 10

Assignment[1]

You may assume for these problems that cycles will appear in the backbone of the list (but you should think about how the problem would be different if the cycle could be in a nested list).

```
(define cycle1 '(a b c))
(set-cdr! (last-pair cycle1) cycle1)

(define cycle2 '(a b c d e))
(set-cdr! (last-pair cycle2) (cddr cycle2))
```

1. Draw cycle1 and cycle2.
2. AS&S problem 3.18. Be sure to test using lists with cycles in different places. For example, the following code does not work for many cases. (Can you figure out which cases it does not work for without running it?)

   ```
   (define (cycle? alist first)

     (define (iter alist)
       (cond ((null? alist) #f)
             ((eq? (car alist) first) #t)
             (else (iter (cdr alist)))))

     (if (null? alist) #f
         (iter (cdr alist))))
   ```

3. One logical extension to the faulty solution above works, but takes $O(n^2)$ time. Design an algorithm that finds cycles in a list using $O(n)$ time. You may find it easier to think of the solution using assignment, but your final code must be strictly functional. Draw a diagram or two and work on this problem alone for at least fifteen minutes. Hint: Think in terms of pointers, and have two pointers - one of which moves twice as fast as the other.

[1] Not *this* assignment, the kind we do in imperative programming.