

Name: \_\_\_\_\_

## Quiz #2 Answers

1. What is the different between flat recursion and deep recursion as far as what they do? How can you tell in the code which one you are dealing with?

Flat recursion only goes through the top elements of a list. Deep recursion goes into all the elements, including those that reside in sublists. The simple way to tell the difference is that flat recursion is like a loop, it calls itself once. Deep recursion is difficult to convert to a loop, it actually needs the power of recursion because it can call itself more than once. In the Scheme flavor of deep recursion it typically calls itself twice, once on the car of the list and once on the cdr.

2. Write code to do a deep replace. This function takes two items and a list. It should return a list where every instance of the first item has been replaced with the second item.

```
(define (deep-replace item1 item2 lst)
  (cond
    ((null? lst) '())
    ((equal? (car lst) item1)
     (cons item2 (deep-replace item1 item2 (cdr lst))))
    ((pair? (car lst))
     (cons (deep-replace item1 item2 (car lst))
           (deep-replace item1 item2 (cdr lst))))
    (else (cons (car lst)
                 (deep-replace item1 item2 (cdr lst))))))
```

Extra Credit: A binary search tree could be implemented in Scheme by using nodes that have three elements: (data left right), where data is what is stored at that node and left and right are the lists representing the subtrees. Remember that in a binary search tree, all data to the left of a node is less than the current nodes data and all data to the right is greater. On the back, write a search method that returns the node that matches a given value.

```
(define (bin-search val node)
  (cond
    ((null? Node) '())
    ((= val (car node)) node)
    ((< val (car node)) (bin-search val (cadr node)))
    (else (bin-search val (caddr node))))
```