

FIFTH QUIZ

You have 15 minutes from the start of class to complete this quiz. Read the problems with care; work with deliberate speed. Don't give us more than we ask for. The usual instructions apply. Good luck!

Problem 1 (4 points)

Fill in the blanks in the function described below. All the parentheses are in the correct places and each blank should be filled by exactly one symbol, function name, or constant. The restaurant structures are defined as usual: (define-struct rrant (name cuisine phone dish price)).

```
;; cheap?: rrant -> boolean
(define cheap?
  (lambda (R)
    (< (rrant-price R) 10.00)))
;; count-cheap-rrants: list-of-rrant -> number
;; Return the number of cheap rrrants in the input list
(define count-cheap-rrants
  (lambda (L)
    (cond
      ((_____ L) _____)

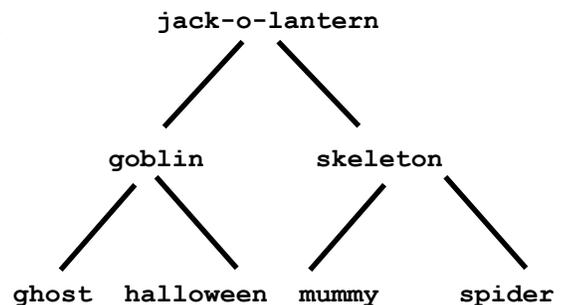
      ((_____ (_____ L))

       _____ 1 (_____ (_____ L))))

    (else (_____ (_____ L))))))
```

Problem 2 (6 points)

(a) (1 point) At the right is a picture of a binary search tree with the items ordered alphabetically. Insert the value "gremlin" into the tree; draw a new branch and node to indicate where it belongs. Be careful to distinguish a left subtree from a right subtree, if necessary (by the angle of the branch).



(b) (1 point) Now insert the value "haunted" into the tree.

(c) (2 points) List all nine items in the tree in the order they would be visited in an inorder traversal of the tree. In other words, if you converted this BST to a list using an inorder traversal, what would be the order of items in the list?

(d) (1 point) In a *preorder* traversal of the original tree (without "gremlin" or "haunted"), what is the very *first* node visited? _____ What is the *second* node visited? _____

(e) (1 point) In a *postorder* traversal of the of the original tree (without "gremlin" or "haunted"), what is the very *first* node visited? _____ What is the *second* node visited? _____

Problem 3 (10 points)

(a) (9 points) Suppose we have a binary search tree of rrant structures (defined as above) with nodes defined as (define-struct node (value left right)), where value is a rrant and left and right are either empty or a node (i.e., a subtree), and the binary search tree property holds. Complete the definition below, adding the necessary code in the blank spaces.

```
;; rrant-name-less?: rrant rrant -> boolean
;; Return true if first rrant's name is alphabetically less than the second
(define rrant-name-less?
  (lambda (R1 R2)
    (string<? (rrant-name R1) (rrant-name R2))))

;; rrant-name-greater?: rrant rrant -> boolean
;; Return true if first rrant's name is alphabetically greater than the second
(define rrant-name-greater?
  (lambda (R1 R2)
    (string>? (rrant-name R1) (rrant-name R2))))

;; add-new-rrant: rrant BST-of-rrants -> BST-of-rrants
;; Insert the input rrant into the input BST according to the alphabetical
;; value of the rrant's name; return the new tree
(define add-new-rrant
  (lambda (R T)
    (cond
      ((empty? T) (make-node
                    ))
      ((rrant-name-less? R (node-value T)) (make-node
                                             ))
      ((rrant-name-greater? R (node-value T)) (make-node
                                                 ))
      (else T))))
```

(b) (1 point) How does this code handle two restaurants with the same name?