FOURTH QUIZ

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

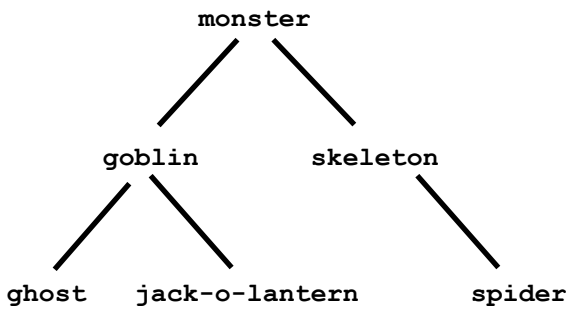
Problem 1 (5 points)

Complete the definition of `remove-italian-restaurants` below. All the parentheses are in the correct places and each blank should be filled by exactly one item: symbol, function name, or constant. Restaurants are defined as usual:

```
(define-struct rrant (name cuisine phone dish price))
;; Italian?: rrant -> Boolean
;; Return true if the input restaurant serves Italian cuisine
(define Italian?
  (lambda (R)
    (string=? 'Italian (rrant-cuisine R))))
;; remove-italian-restaurants:  list-of-rrants -> list-of-rrants
;; Return a list containing all the restaurants on the input list that
;; do NOT serve Italian cuisine.
(define remove-italian-restaurants
  (lambda (L)
    (cond

      ((empty? _______________) _______________)

      ((_______________ (_______________ L)) (_______________ (_______________ L)))

      (else (_______________ (_______________ L) (_______________ (_______________ L)))))))
```

Problem 2 (5 points)

(a) (1 point) At the right is a picture of a binary search tree. Insert the value "pumpkin" 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 "witch" into the tree.

(c) (2 points) List all eight 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) (1/2 point) In a *preorder* traversal of the original tree (without "pumpkin" or "witch"), what is the very *last* node visited?

(e) (1/2 point) In a *postorder* traversal of the of the original tree (without "pumpkin" or "witch"), what is the very *first* node visited?

Problem 3 (10 points)

Suppose we have a binary search tree of `rrant` structures (defined as above), with nodes defined as follows:

```
(define-struct node (key value left right))
```

where `key` is the `rrant`'s name (a string), `value` is a `rrant`, `left` and `right` are either empty or a node, and the binary search tree property applies. Complete the definition below of `find-named-rrant`, adding the necessary code to the blank spaces.

```
;; find-named-rrant:  string  BST-of-rrants  ->  rrant
;; If the string matches the name of a rrant in the tree, return that rrant.
;; Otherwise, return empty.
(define find-named-rrant
  (lambda (n T)
    (cond
      ((empty?



      ((string=?



      ((string<?



      (else
```