Informatics 41 • Fall 2007 • David G. Kay • UC Irvine Your name ________________________

FOURTH 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 (8 points)

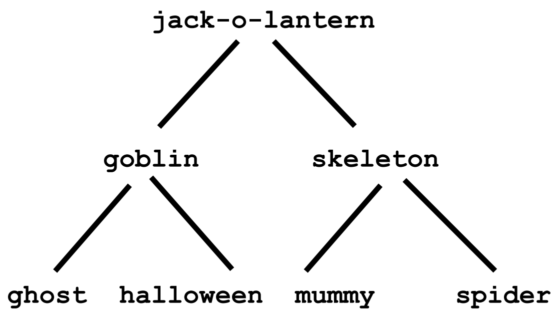
Complete the definition of `collect-expensive-rrants` below.

```
;; expensive?: rrant -> boolean
;; Return true if the input rrant's price is over $20.
(define expensive?
  (lambda ( R )
    (> (rrant-price R) 20)))

;; collect-expensive-rrants:  list-of-rrant -> list-of-rrant
;; Return a list of all the expensive restaurants on the input list
```

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

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

Problem 3 (7 points)

Suppose we have a binary search tree of numbers, with nodes defined as follows:

```
(define-struct BSTnode (rootvalue left right))
```

where `rootvalue` is a number, `left` and `right` are either empty or a node, and the binary search tree property applies. Fill in the blanks below to complete the definition below of `traverse-in-order`; each blank should be filled by exactly one item: symbol, function name, or constant.

```
;; traverse-in-order:  BST-of-numbers  ->  list-of-numbers
;; Return a list of all the numbers in the input tree, in order
(define traverse-in-order
  (lambda (T)
    (cond
      ((____________________  ____________________) ____________________)


      (else (append

                  (____________________ (____________________ T))


                  (list (____________________   ____________________))


                  (____________________ (____________________ T)))))))
```