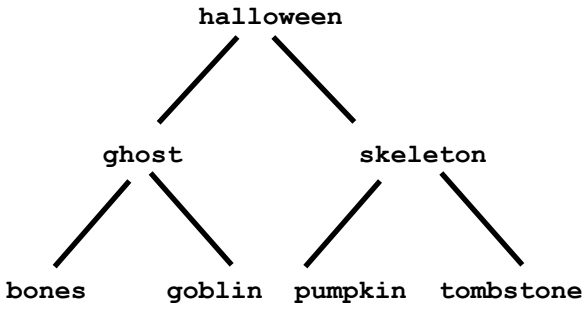
SIXTH QUIZ

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 (5 points)

(a) (1 point) At the right is a picture of a binary search tree with the items ordered alphabetically. Insert the value "black-cat" 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 "spider" into the tree.

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

(d) (1/2 point) In a *pre-order* traversal of the original tree (without "black-cat" or "spider"), what is the very *first* node processed?

(e) (1/2 point) In a *post-order* traversal of the of the original tree , what is the very *last* node processed?

Problem 2 (15 points)

Your electronic cookbook contains a list of recipes as defined in a previous lab assignment:

```
(define-struct recipe (title ingredients steps))
```

where the title is a symbol, ingredients is a list of symbols, and steps is a list of steps (where each step is a list of symbols); for example:

```
(make-recipe 'ThaiIcedCoffee
  '(coffee sugar condensed-milk ice)
  '((brew coffee) (mix with sugar and condensed-milk) (pour coffee mixture over ice)))
```

On this quiz, we do not expect you to use `map`, `filter`, or `foldr`, but you may use them if you're confident enough to let your score depend on it.

(a) (2 points) Define the function `recipe-contains-ingredient?` as described below; it just takes one line if you use the function `member?`.

```
;; recipe-contains-ingredient?:  recipe  symbol  ->  boolean
;; Return true if the symbol occurs in the recipe's ingredients list
(define recipe-contains-ingredient?
  (lambda (rec ing)
```

(b) (2 points) Define the function `step-short?` described below; using `length` you can do it in one line.

```
;; step-short?:  list-of-symbols(one recipe step)  ->  boolean
;; Return true if input list (one recipe step) has 6 symbols (words) or fewer
(define step-short?
  (lambda (step)
```

(c) (3 points) Define the function `all-steps-short?` as described below. For full credit, use `step-short?` as described above (whether or not you defined it correctly).

```
;; all-steps-short?: list-of-steps -> boolean
;; Return true if input list is empty or each step on list is short (6 words or fewer)
(define all-steps-short?
  (lambda (LOS)
   (cond
     ((empty? LOS) _________________)

     ((_________________ (first LOS)) (_________________ (rest LOS)))

     (else _________________))))
```

(d) (3 points) Define the function `recipe-simple?` as described below. For full credit, use `all-steps-short?` where appropriate.

```
;; recipe-simple?:  recipe  ->  boolean
;; Return true if recipe has under 10 steps and each step is short (6 words or fewer)
(define recipe-simple?
  (lambda (rec)
    (and
      (_________________ (_________________ (_________________ rec)) 10)

      (_________________ (_________________ rec)))))
```

(e) (5 points) The first word (symbol) of each step is a verb (e.g., bake, mix, grill); we can call that the *technique* involved in that step. Assume this function is already defined; you don't have to define it now:

```
;; recipe-involves-technique?:  recipe  symbol(technique)  ->  boolean
;; Return true if any of the recipe's steps involves the technique (second argument).
```

Now, suppose you want to select recipes that will help you practice a particular technique. Define the function `simple-practice-recipes` as described below, using previously defined functions where appropriate:

```
;; simple-practice-recipes:  list-of-recipe  symbol(technique)  ->  list-of-recipe
;; Return all recipes on the input list that are both simple and involve
;; the specified technique.
(define simple-practice-recipes
  (lambda (cookbook technique)
    (cond
```