You should provide a signature/purpose for any helper function you define.

Your programs may contain only the following DrRacket constructs:

```racket
define define-struct cond else if
```

and the following primitive operators:

```racket
empty? cons? cons first rest list append
+ - * / = < > <= >=
string=? string-length symbol=?
```

recognizers for any defined data types

```racket
and or not
```

and the operators introduced by `define-struct`.

You may, of course, use whatever constants are necessary (`empty`, `true`, `false`, `0`, etc.)
1. **(Lists of structs: 30 points)**

Here are a set of data definitions that could have been used in Homework 4, where you wrote programs for a list of borrowers for a microlending institution:

```scheme
;; Borrower is a (make-borrower String String String Number Number)
(define-struct borrower (name country business amount raised))
;; interp: amount is the amount of the requested loan, in dollars
;; raised is the percentage of the loan that has been raised so far
;; (i.e. it is a number between 0 – 100 inclusive)

;; a ListOfBorrower is one of
;;   empty
;;   (cons Borrower ListOfBorrower)
```

Write a function to satisfy the following signature and purpose. You **must** develop a helper function and use the helper in your solution. Provide a signature and purpose for your helper. (You may continue your answer on the next page if you need more space.)

```scheme
;; small-loans: ListOfBorrower String -> ListOfBorrower
;; consumes a list of borrowers and the type of business and produces
;; a list of the borrowers in the given business who have loan requests of
;; less than $1000
```
(additional page for Problem 1)
2. (Trees: 25 points)
   Here are a set of data definitions:

   ;; Item is a (make-item String Number)
   (define-struct item (name price))
   ;; interp: represents an item for sale in a store, where
   ;;      name is the name of the item
   ;;      price is the price of the item in dollars

   ;; ListOfItem is one of
   ;;      empty
   ;;      (cons Item ListOfItem)

   ;; TreeList is one of
   ;;      'unknown
   ;;      (make-listnode ListOfItem TreeList TreeList)
   (define-struct listnode (list left right))

   (a) (5 points) Provide the template for functions that operate on Item. (You don’t have to write the signature/purpose.)
(b) (5 points) Provide the template for functions that operate on `ListOfItem`. (You don’t have to write the signature/purpose.)

(c) (15 points) Provide the template for functions that operate on `TreeList`. (You don’t have to write the signature/purpose.)
3. **Hierarchies (25 points)** The following data definitions are used to represent the organizational structure at a manufacturing company:

```scheme
;; Employee is a (make-employee String String ListOfEmployee)
(define-struct employee (name position subordinates))
;; interp: represents an employee with
;;       a name,
;;       position (such as CEO or engineer)
;;       a list of all direct subordinates

;; ListOfEmployee is one of
;;   empty
;;   (cons Employee ListOfEmployee)
```

Write a function (or functions) to satisfy the following signature/purpose:

```scheme
;; has-no-subordinates: Employee -> ListOfString
;; produces a list of the names of every employee in the hierarchy
;; who has no one reporting to them
```

(You may continue your answer on the next page if you need more space.)
(additional page for Problem 3)
4. (20 points) A binary search tree (BST) of numbers can be defined as:

;; BST is one of
;; 'unknown
;; (make-node Number BST BST)
(define-struct node (key smaller larger))
;; interp: a binary search tree where for each node n in the tree,
;; the key values in n’s left subtree are smaller than the key for n,
;; the key values in n’s right subtree are greater than the key for n

(a) (5 points) Here’s a picture of an incomplete binary search tree:

![Binary Search Tree Diagram]

What range of key values would be allowed for the node containing the ???’s?
Here's function that operates on a BST:

```scheme
;; in-tree?: BST Number -> Boolean
;; returns true if the given number is in the binary search tree
(define (in-tree? abst num)
  (cond [(symbol? abst) false]
        [(node? abst) (or (= (node-key abst) num)
                          (in-tree? (node-smaller abst) num)
                          (in-tree? (node-larger abst) num))])
)
```

(b) (5 points) Does the function as written satisfy the signature/purpose? (Answer yes or no)

(c) (10 points) The function as written does not exploit the binary search tree property. Write a function that both

- satisfies the given signature/purpose, and
- solves the problem in such a way that the binary search tree property is used to more efficiently execute the function