NAME: SECTION:

CS 1101
Exam 2
A-Term 2014

Question 1:	_____	(25)
Question 2:	_____	(25)
Question 3:	_____	(25)
Question 4:	_____	(25)
TOTAL:	_____	(100)

You have 50 minutes to complete this exam. You do not need to show templates (unless a problem states otherwise), but you may receive partial credit if you do. You also do not need to show test cases or examples of data definitions (unless a problem states otherwise), but you may develop them if they will help you write the programs.

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

Your programs may contain only the following DrRacket constructs:

define define-struct cond else if

and the following primitive operators:

empty? cons? cons first rest list append

`+ - * / = < > <= >=`

string=? string-length symbol=?

recognizers for any defined data types

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. (25 points) Use the following data definitions for Problem 1:

```
(define-struct song (title artist time price))
;; A Song is a (make-song String String Natural Number)
;; interp:   a song in a playlist where
;;             title is the title of the song
;;             artist is the name of the artist
;;             time is the length of the song, in seconds
;;             price is the price of the song

;; a Playlist is one of
;;    empty
;;    (cons Song Playlist)
```

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.)

```
;; long-by-artist?:  Playlist String Natural -> Boolean
;; produces true if there exists a song in the playlist which is by the given
;; artist and which has a length longer than the given number of seconds
```

(additional page for Problem 1)

2. (25 points) A binary search tree (BST) of numbers can be defined as:

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

(a) (5 points) Draw a picture of a BST that contains the numbers
1, 3, 5, 7, 9, 11, 13, 15
Your BST should have as few levels as possible.

(b) (5 points) Here's function that operates on a BST:

```
(define (mystery abst anum)
  (cond [(false? abst) (make-node anum false false)]
        [(node? abst)
             (cond [(< anum (node-key abst))
                        (make-node (node-key abst)
                                   (mystery (node-left abst) anum)
                                   (node-right abst))]
                   [else (make-node (node-key abst)
                                    (node-left abst)
                                    (mystery (node-right abst) anum))])]))
```

In one clear, concise sentence, explain what the function mystery does. (This question is worth only 5 points; don't spend a lot of time on it).

(c) (15 points) Write a function to satisfy the following signature and purpose:

```
;; any-larger?:  BST Number -> Boolean
;; produces true if the given BST contains any values greater than
;; the given number
```

Your function should be written efficiently (the solution should take advantage of the binary search tree invariant).

3. (25 points) The following data definitions can be used to represent information about a person and his/her descendants:

```
(define-struct person (name age female? children))
;; a Person is a (make-person String Natural Boolean ListOfPerson)
;; interp:  represents a person with
;;          name as the person's name
;;          age as the person's age (in years)
;;          female? is true if the person is a female, false if
;;                  the person is a male
;;          children as the person's list of children

;; ListOfPerson is one of
;;   empty
;;   (cons Person ListOfPerson)
```

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

```
;; any-women-named?:  Person String -> Boolean
;; produces true if the tree contains any females who have the given name
```

(You may continue your answer on the next page if you need more space.)

(additional page for Problem 3)

(there's one more problem on Page 8)

4. (25 points) The members of a soccer team have set up a telephone call tree, so that in the event that a game is canceled, everyone on the team can be notified in a timely manner. The call tree is set up in such a way that a person will never be required to call more than three other people.

   Usually we start with a data definition and create an example using the data definition. In this problem, you'll work the other way. You'll first be given an example of data. Your job is to come up with a complete data definition (or set of data definitions) that fits the example.

   Here's the example:

```
(define call-tree
  (make-caller "Joe" 5088315555
               (make-caller "Ed" 5088314456 false false false)
               (make-caller "May" 9789787789 false false false)
               (make-caller "Sally" 6172078455
                            (make-caller "Drew" 5088315567 false false false)
                            (make-caller "Xi" 5088318678 false false false)
                            false)))
```

   Provide a complete (set of) data definition(s) to fit the example. Your data definition should include appropriate types and comments (including interpretations). Your data definition should allow the given example to be created by Racket without causing any errors.

(end of exam)