CMPE 462 Assignment 3

(Haskell)

Due: 04 November 2019 (Monday), beginning of the lab session

To be done in groups of two. Pick your partner!

Take the following definitions as your starting point. A "Tree" object represents a binary search tree.

```
data Tree a = Empty | Node (Tree a) a (Tree a) deriving (Show)

insert n Empty = Node Empty n Empty
insert n (Node left value right) =
    if
        n<value
    then
        (Node (insert n left) value right)
    else
         (Node left value (insert n right))

insertList [] = Empty
insertList (h:t) = insert h (insertList t)

size Empty = 0
size (Node left value right) = 1 + (size left) + (size right)

t = Node (Node Empty 5 Empty) 10 (Node Empty 15 Empty)
```

Note how data constructors need to be put in parenthesis! Also please note that we need **deriving (Show)** after the type declaration. We need this in order to see the resulting tree.

Implement the following operations on binary search trees.

find v t = True if value v is in the tree, otherwise False

inorder t = a list containing the values in the tree in the same order as in an "inorder" traversal.

inner_nodes t = a list containing the values in the inner (non-leaf) nodes of the tree.

leaves t = a list containing the values in leaf nodes of the tree.

subset t1 t2 = all values in tree t1 are also in tree t2

union t1 t2 = the binary search tree that contains all the values in t1 and t2 *without duplication!*

difference t1 t2 = the list of values that are in t1 but not in t2

max t1 = the maximum value in t1

min t1 = the maximum value in t1

sum t1 = the sum of values in t1

BONUS:

tree_zip t1 t2 = a tree whose node values are **pairs** of corresponding values from t1 and t2. Must work similarly to the zip function for lists. If a node in t1 (t2) does not have a corresponding node in t2 (t1), then that node must not be represented in the result.