Binary Search Trees

```
--------------------------------------------------------------------------------

data BST a = EmptyBST | Node ( BST a ) a ( BST a ) deriving Show

--------------------------------------------------------------------------------

-- leaf v : the single-node BST storing the value ’v’

leaf :: a -> BST a

leaf v = Node EmptyBST v EmptyBST

--------------------------------------------------------------------------------

-- numNodes bst : the number of nodes in the BST ’bst’

numNodes :: BST a -> Int

numNodes EmptyBST             = 0

numNodes ( Node lSub _ rSub ) = numNodes lSub + 1 + numNodes rSub

--------------------------------------------------------------------------------

-- insert v bst : the BST formed by inserting the value ’v’ into the BST ’bst’

insert :: Ord a => a -> BST a -> BST a

insert v EmptyBST                = leaf v

insert v ( Node lSub root rSub ) = if v < root then
                                      Node ( insert v lSub ) root rSub
                                   else
                                      Node lSub root ( insert v rSub )

--------------------------------------------------------------------------------

-- sameShape bst1 bst2 : do the BSTs ’bst1’ and ’bst2’ have the same shape ?

sameShape :: BST a -> BST b -> Bool

sameShape EmptyBST               EmptyBST               = True

sameShape ( Node lSub1 _ rSub1 ) ( Node lSub2 _ rSub2 ) = sameShape lSub1 lSub2
                                                          &&
                                                          sameShape rSub1 rSub2

sameShape _                      _                      = False

--------------------------------------------------------------------------------

t1 :: BST Int

t1 = Node ( Node ( leaf 1 ) 2 ( leaf 3 ) )
            4
            ( Node EmptyBST 5 ( Node ( leaf 6 ) 7 EmptyBST )
          )

--------------------------------------------------------------------------------
```

```
--------------------------------------------------------------------------------

t2 :: BST Char

t2 = insert ’F’
            ( insert ’A’
                     ( insert ’C’
                              ( insert ’G’
                                       ( insert ’B’
                                                ( insert ’E’
                                                         ( insert ’D’
                                                                  EmptyBST
                                                         )
                                                )
                                       )
                              )
                     )
            )

--------------------------------------------------------------------------------
--------------------------------------------------------------------------------

> numNodes t1
7

> numNodes t2
7

> numNodes EmptyBST
0

> numNodes ( leaf "green" )
1

> numNodes ( Node ( leaf "blue" ) "green" ( leaf "red" ) )
3

> numNodes ( Node ( leaf ( \n -> n+1 ) ) ( \n -> n*n ) ( leaf ( \n -> n-1 ) ) )
3

> sameShape t1 t2
True

> sameShape ( insert 8 t1 ) ( insert ’H’ t2 )
True

> sameShape ( insert 4 t1 ) ( insert ’H’ t2 )
False

--------------------------------------------------------------------------------
```