

# Isabelle/HOL Exercises

## Trees, Inductive Data Types

### Tree Traversal

Define a datatype `'a tree` for binary trees. Both leaf and internal nodes store information.

```
datatype 'a tree = Tip "'a" | Node "'a" "'a tree" "'a tree"
```

Define the functions `preOrder`, `postOrder`, and `inOrder` that traverse an `'a tree` in the respective order.

```
preOrder :: "'a tree ⇒ 'a list"
postOrder :: "'a tree ⇒ 'a list"
inOrder  :: "'a tree ⇒ 'a list"
```

**primrec**

```
"preOrder (Tip a)          = [a]"
"preOrder (Node f x y) = f#((preOrder x)@(preOrder y))"
```

**primrec**

```
"postOrder (Tip a)          = [a]"
"postOrder (Node f x y) = (postOrder x)@(postOrder y)@[f]"
```

**primrec**

```
"inOrder (Tip a)          = [a]"
"inOrder (Node f x y) = (inOrder x)@[f]@(inOrder y)"
```

Define a function `mirror` that returns the mirror image of an `'a tree`.

```
mirror :: "'a tree ⇒ 'a tree"
```

**primrec**

```
"mirror (Tip a)          = (Tip a)"
"mirror (Node f x y) = (Node f (mirror y) (mirror x))"
```

Suppose that `xOrder` and `yOrder` are tree traversal functions chosen from `preOrder`, `postOrder`, and `inOrder`. Formulate and prove all valid properties of the form `xOrder (mirror xt) = rev (yOrder xt)`.

```
theorem "preOrder (mirror xt) = rev (postOrder xt)"
  apply (induct_tac xt)
```

```
  apply auto
done
```

```
theorem "postOrder (mirror xt) = rev (preOrder xt)"
  apply (induct_tac xt)
  apply auto
done
```

```
theorem "inOrder (mirror xt) = rev (inOrder xt)"
  apply (induct_tac xt)
  apply auto
done
```

Define the functions *root*, *leftmost* and *rightmost*, that return the root, leftmost, and rightmost element respectively.

```
root  :: "'a tree ⇒ 'a"
leftmost :: "'a tree ⇒ 'a"
rightmost :: "'a tree ⇒ 'a"
```

```
primrec
  "root (Tip a)          = a"
  "root (Node f x y) = f"
```

```
primrec
  "leftmost (Tip a)          = a"
  "leftmost (Node f x y) = (leftmost x)"
```

```
primrec
  "rightmost (Tip a)          = a"
  "rightmost (Node f x y) = (rightmost y)"
```

Prove or disprove (by counterexample) the theorems that follow. You may have to prove some lemmas first.

```
lemma [simp]: "inOrder xt ~ = []"
  apply (induct_tac xt)
  apply auto
done
```

```
lemma [simp]: "ys ~ = [] → last (xs @ ys) = last ys"
  apply (induct_tac xs)
  apply auto
done
```

```
theorem "last (inOrder xt) = rightmost xt"
  apply (induct_tac xt)
  apply auto
done
```

```
theorem "hd (inOrder xt) = leftmost xt"
  apply (induct_tac xt)
  apply auto
done
```

```
theorem "hd (preOrder xt) = last (postOrder xt)"
  apply (induct_tac xt)
  apply auto
done
```

```
theorem "hd (preOrder xt) = root xt"
  apply (induct_tac xt)
  apply auto
done
```

```
theorem "hd (inOrder xt) = root xt"
  quickcheck
:
```

Counterexample found:  
xt = Node 1 (Tip 0) (Tip 0)

```
theorem "last (postOrder xt) = root xt"
  apply (induct_tac xt)
  apply auto
done
```