Isabelle Exercises

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

```
datatype 'a tree
```

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

```
consts preOrder :: "'a tree => 'a list"
consts postOrder :: "'a tree => 'a list"
consts inOrder :: "'a tree => 'a list"
```

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

```
consts mirror :: "'a tree => 'a tree"
```

- Suppose that `xOrder` and `yOrder` are tree traversal functions chosen from the above. Formulate and prove all valid properties of the following form:

```
xOrder (mirror xt) = rev (yOrder xt)
```

(Here `rev` is a predefined function on lists, reversing a list.)

- Define the functions `root`, `leftmost`, and `rightmost`, that return the root, the leftmost, and the rightmost element of `'a tree` respectively.

```
consts root :: "'a tree => 'a"
consts leftmost :: "'a tree => 'a"
consts rightmost :: "'a tree => 'a"
```

- Prove (or disprove using `quickcheck`) the theorems that follow. You may/will have to prove some lemmas first.

```
theorem "last (inOrder xt) = rightmost xt"
theorem "hd (inOrder xt) = leftmost xt"
theorem "hd (preOrder xt) = last (postOrder xt)"
theorem "hd (preOrder xt) = root xt"
theorem "hd (inOrder xt) = root xt"
theorem "last (postOrder xt) = root xt"
```