Tree Traversal

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 traverse an \('a\ tree\) in the respective order.

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

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

```
mirror :: "'a tree ⇒ 'a tree"
```

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

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"
```

Prove (or let Isabelle disprove) the theorems that follow. You may 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"
```