Exercises E6

1. The data type Tree may be parameterised by an arbitrary type (the Tree data type in the lecture slides on Data Decls (II) is the same as Tree Int):

   ```
   data Tree a = Leaf a | Node (Tree a) a (Tree a)
   ```

   Define the following predicate to test if a value occurs in a search tree:

   ```
   occurs :: Ord a => a -> Tree a -> Bool
   ```

2. In the Haskell library, one can find the following data type:

   ```
   data Ordering = LT | EQ | GT
   ```

   together with a function

   ```
   compare :: Ord a => a -> a -> Ordering
   ```

   that decides if one value in an ordered type is less than (LT), equal to (EQ), or greater than (GT) another value.

   (1) Using this function, redefine the function occurs for search trees, as defined in the above exercise.
   (2) Why is this new definition more efficient than the version defined in the above exercise?