1. **Binary Trees.** Write a function `reverse` which creates the mirror image of a binary tree. That is, if $T$ is a binary tree, then $\text{reverse}(T)$ is a binary tree in which, at every node, left and right branches are interchanged.

A binary tree is said to be *balanced* if for each node $\text{Br}(x, t_1, t_2)$ the sizes of $t_1$ and $t_2$ differ by at most one. Write a function `balanced` of type `'a tree -> bool` which determines whether a tree is balanced. One obvious solution involves checking the size of every subtree, but this is inefficient because it repeats a lot of computation. Can you do this more efficiently?

2. **Arrays** Write a function that takes an array in binary tree form and returns a list of the elements of the array, in order. Can you do this efficiently, i.e. without extracting each element by looking up the subscript?

3. **Merge Sort** Write a generic version of `mergesort`, which takes a comparison function as argument.

4. **Minimum** Write a functional to compute the minimum value $\min_{i=0}^{n-1} f(i)$ of a function $f$. Use the functional to express the two dimensional minimum $\min_{i=0}^{n-1} \min_{j=0}^{n-1} g(i, j)$ of a function $g$ of two arguments.