REKORSHION

```
maximum' :: (Ord a) => [a] -> a
maximum' [] = error " maximum of empty list"
maximum' [x]  = x
maximum' (x:xs)
   | x > maxTail     = x
   | otherwise       = maxTail
   where maxTail = maximum' xs
```

```
maximum' :: (Ord a) => [a] -> a
maximum' []   = error "maximum of empty list"
maximum' [x]  = x
maximum' (x:xs) = max x (maximum' xs)
```

$\overbrace{\text{(Num i, Ord i)}}^{\text{addition + companision}}$

```
replicate' :: (Num i, Ord i) => i -> a -> [a]
replicate' n x
   | n <= 0        = []
   | otherwise     = x:replicate' (n-1) x
```

```
take' :: (Num i, Ord i) => i -> [a] -> [a]
take' n _
   | n <= 0        = []
take' _ []         = []
take' n (x:xs)     = x : take' (n-1) xs
```

```
reverse' :: [a] -> [a]
reverse' []     = []
reverse' (x:xs) = ~~xs++[x]~~ reverse' xs ++ [x]
```

```
repeat' :: a -> [a]
repeat' x = x : repeat' x

zip' :: [a] -> [b] -> [(a,b)]
zip' _ [] = []
zip' [] _ = []
zip' (x:xs) (y:ys) = (x,y) : zip' xs ys

elem' :: (Eq a) => a -> [a] -> Bool
elem' _ [] = False
elem' a (x:xs)
    | a == x    = True
    | otherwise = a `elem'` xs

quicksort :: (Ord a) => [a] -> [a]
quicksort [] = []
quicksort (x:xs) =
    let smallerSorted = quicksort [a | a <- xs, a <= x]
        biggerSorted = quicksort [a | a <- xs, a > x]
    in smallerSorted ++ [x] ++ biggerSorted

λ quicksort [10, 2, 5, 3, 1, 6, 7, 4, 2, 3, 4, 8, 9]
[1, 2, 2, 3, 3, 4, 4, 5, 6, 7, 8, 9, 10]
λ quicksort "the quick brown fox jumps over the lazy dog"
"    abcdeeefghhijklmnoooopqrrsttuuvwxyz"
```