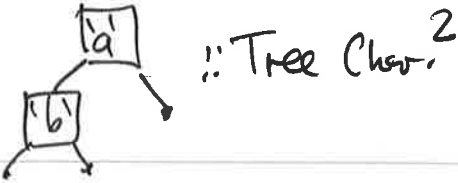
Lecture 6.

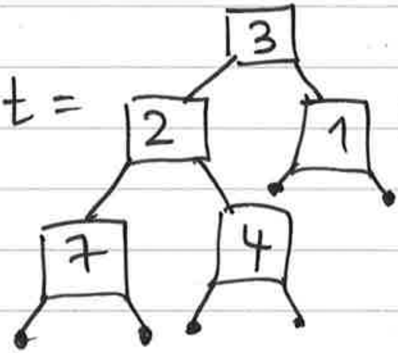
```
isort :: Ord a => [a] -> [a]
isort xs = foldr f k
  where
```

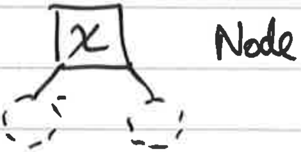
```
isort [2, 1, 4, 3]
isort (2 : 1 : 4 : 3 : [])

f 2 (f 1 (f 4 ((f 3 k))))
                [3]
              = f 4 [3]
                [3,4]
        = f 1 [3,4]
          [1,3,4]
  = f 2 [1,3,4]
    [1, 2, 3, 4]
                                  k = []
```

```
f :: Ord a => a -> [a] -> [a]
f x [] = [x]
f x (y:ys) | x < y = x : y : ys
           | otherwise = y : f x ys
```

Trees







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

```
Node (Node (Node Tip 7 Tip) 2 (Node Tip 4 Tip)
     3  (Node Tip 1 Tip)
```

```
size :: Tree a → Int
size Tip = 0
size (Node l x r) = 1 + size l
                      + size r
```

```
height :: Tree a -> Int
height Tip = 0
height (Node l x r) =
   max (height l) (height r) + 1
```

```
flatten t = [7, 2, 4, 3, 1]
```

```
flatten :: Tree a -> [a]
flatten Tip = []
flatten (Node l x r) =
```

$$\underbrace{\text{flatten } l}_{[a]} + + [x] + + \underbrace{\text{flatten } r}_{[a]}$$

alternative:

```
   flatten l ++ x : flatten r
```

```
               (Ord a, Eq a) =>
member :: Tree a -> a -> Bool
member Tip x = False
member (Node l y r) x =
     | x == y     = True
     | x < y      = member l x
     | otherwise  = member r x
```

```
             Ord a =>
mkTree  :: [a] -> Tree a
mkTree  []     = Tip
mkTree  (x:xs) = Node l x r
    where  l = mkTree ys
           r = mkTree zs
           (ys, zs) = partition x xs

                       a ->
partition :: Ord a => [a] -> ([a],[a])
partition x []  = ([],[])
partition x  ys = ([y | y ← ys, y ≤ x]
                  , [z | z ← xs, z > x])
          Ord a =>
qsort :: [a] -> [a]
qsort  xs  = flatten (mkTree xs)
```