Lecture 9

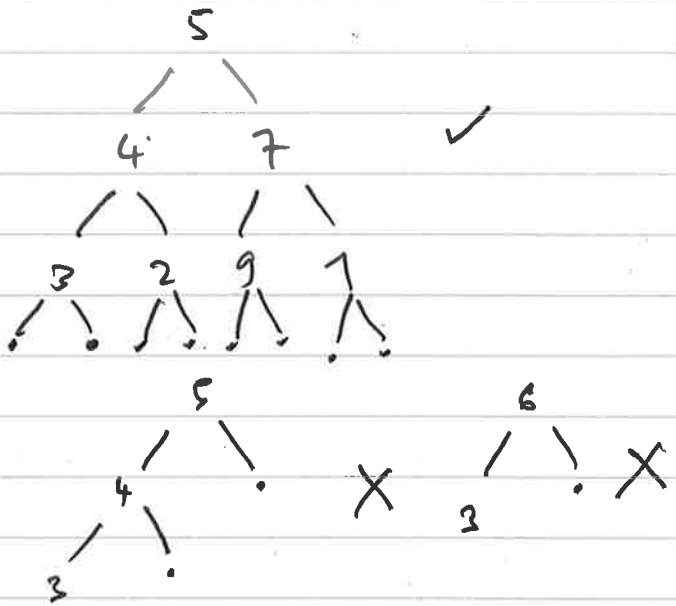
```
data Tree a = Tip
            | Node (Tree a) a (Tree a)

Tip :: Tree a
Node :: Tree a → a → Tree a → Tree a

foldTree :: b → (b → a → b → b) → Tree a → b
foldTree tip node Tip = tip
foldTree tip node (Node l x r) =
    node (foldTree tip node l)
         x
         (foldTree tip node r)

sumTree :: Tree Int → Int
sumTree = foldTree tip node
  where tip :: Int
        tip = 0
        node :: Int → Int → Int → Int
        node l x r = l + x + r
```

~~size~~

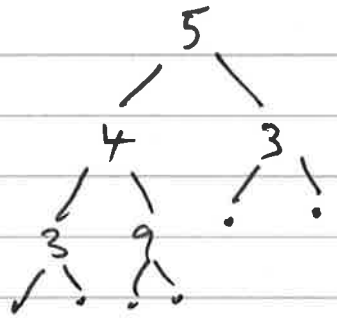
~~height Tree a~~

```
height :: Tree a → Int
height = foldTree tip node
  where
    tip :: Int
    tip = 0
    node :: Int → a → Int → Int
    node l x r = 1 + max l r
```

Perfect tree:



```
perfect :: Tree a → Bool
perfect = foldTree tip node
  where
    tip :: Bool
    tip = True
    node :: Bool → a → Bool → Bool
    node l x r =
```



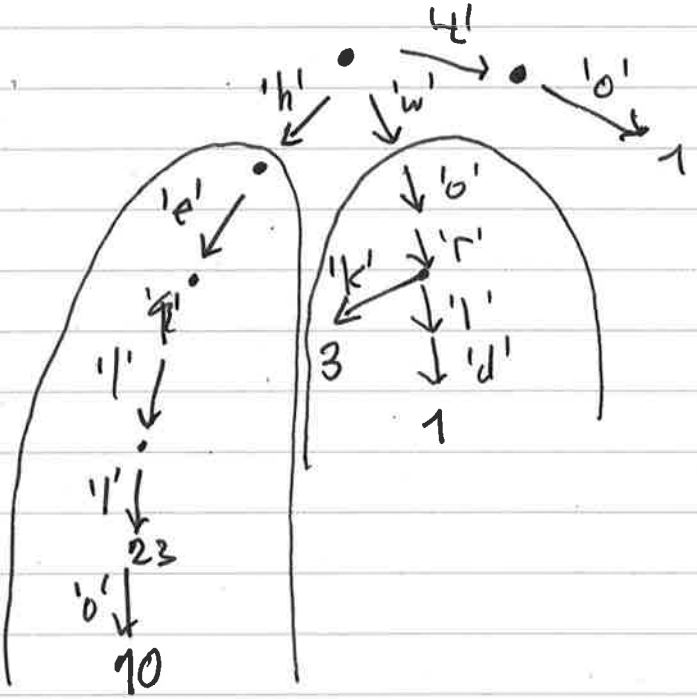

```
fst :: (a,b) → a
snd :: (a,b) → b
```

Try again:

```
perfect :: Tree a → (Bool, Int)
perfect = foldTree tip node
  where
    tip :: (Bool, Int)
    tip = (True, 0)
    node :: (Bool, Int) → a → (Bool, Int) → (Bool, Int)
    node l x r = (fst l && fst r && snd l == snd r
                  , 1 + max (snd l) (snd r))
```

Trie Structure

"hello"



Map

m_0 = empty $\quad \emptyset \quad$ :: Map Char Int

m_1 = insert 'h' 3 empty. $\quad$ 'h' $\mapsto$ 3 :: Map Char Int

m_2 = insert 'w' 20 m_1 $\quad$ ('h' $\mapsto$ 3 :: Map Char Int
'w' $\mapsto$ 20)

member :: k → Map k v → Bool

member 'h' m_2 == True
member 'j' m_2 == False

(!) :: Map k v → k → v

m_2 ! 'w' = 20
m_2 ! 'z' = error "!"

```
data Trie a = Trie Int (Map a (Trie a))

emptrie :: Trie a
emptrie = Trie 0 empty


count :: [a] -> Trie a -> Int
count [] (Trie n kvs) = n
count (k:ks) (Trie n kvs)
    | member k kvs  = count ks (kvs ! k)
    | otherwise     = 0

tally :: [a] -> Trie a -> Trie a
tally [] (Trie n kvs) = Trie (n+1) kvs
tally (k:ks) (Trie n kvs)
    | member k kvs = Trie n (insert k (tally ks (kvs ! k)) kvs)
    | otherwise    = Trie n (insert k (tally ks emptrie)
                                      kvs)
```