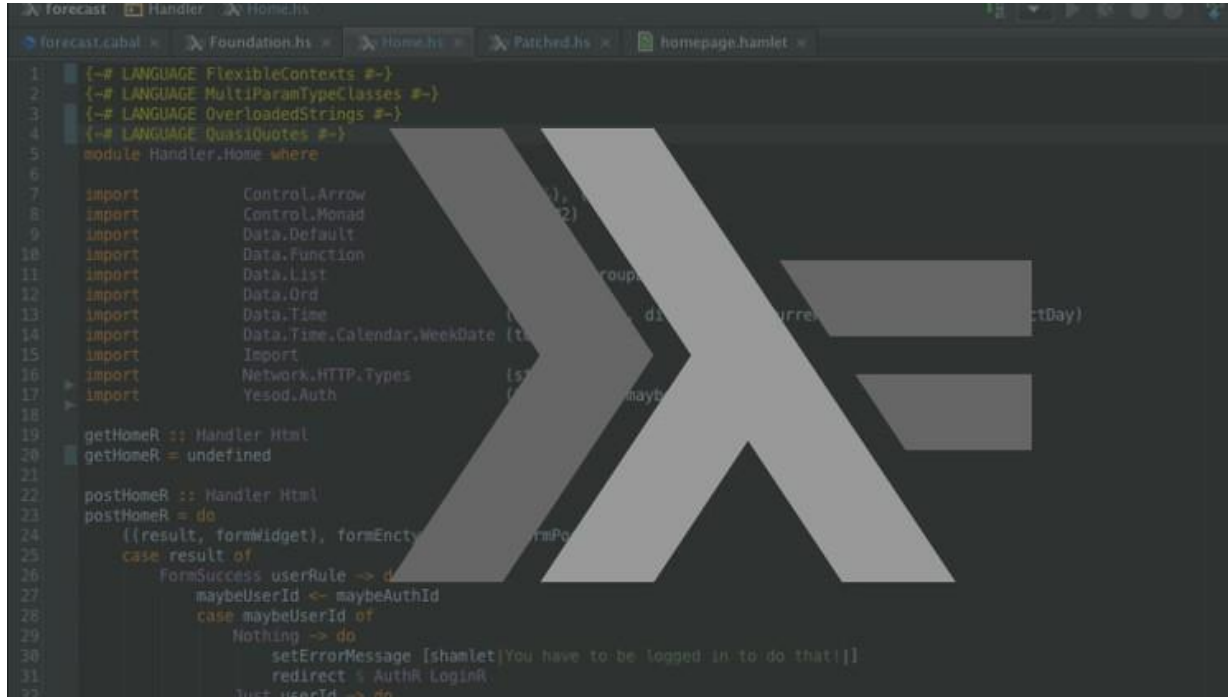


# Multiway Tree Problems



```
1 {-# LANGUAGE FlexibleContexts #-}
2 {-# LANGUAGE MultiParamTypeClasses #-}
3 {-# LANGUAGE OverloadedStrings #-}
4 {-# LANGUAGE QuasiQuotes #-}
5 module Handler.Home where
6
7 import Control.Arrow
8 import Control.Monad
9 import Data.Default
10 import Data.Function
11 import Data.List
12 import Data.Ord
13 import Data.Time
14 import Data.Time.Calendar.WeekDate
15 import Import
16 import Network.HTTP.Types
17 import Yesod.Auth
18
19 getHomeR :: Handler Html
20 getHomeR = undefined
21
22 postHomeR :: Handler Html
23 postHomeR = do
24   ((result, formWidget), formEnctype) <- runFormPost
25   case result of
26     FormSuccess userRule -> do
27       maybeUserId <- maybeAuthId
28       case maybeUserId of
29         Nothing -> do
30           setErrorMessage [shamlet|You have to be logged in to do that!|]
31           redirect % AuthR.LoginR
32         Just userId -> do
```

# Problem 1



```
forecast.cabal x Handler x Home.hs
forecast.cabal x Foundation.hs x Home.hs x Patched.hs x homepage.hamlet x
1 {-# LANGUAGE FlexibleContexts #-}
2 {-# LANGUAGE MultiParamTypeClasses #-}
3 {-# LANGUAGE OverloadedStrings #-}
4 {-# LANGUAGE QuasiQuotes #-}
5 module Handler.Home where
6
7 import Control.Arrow ((>>=), (***))
8 import Control.Monad (replicateM2)
9 import Data.Default (Default)
10 import Data.Function (on)
11 import Data.List (group)
12 import Data.List (Data.List)
13 import Data.List (Data.List)
14 import Data.List (Data.List)
15 import Data.List (Data.List)
16 import Data.List (Data.List)
17 import Data.List (Data.List)
18
19 getHomeR :: Handler Html
20 getHomeR = undefined
21
22 postHomeR :: Handler Html
23 postHomeR = do
24   ((result, formWidget), formEnctype, formPost) <- runFormPost
25   case result of
26     FormSuccess userRule -> do
27       maybeUserId <- maybeAuthId
28       case maybeUserId of
29         Nothing -> do
30           setErrorMessage [shamlet|You have to be logged in to do that!|]
31           redirect % AuthR.LoginR
32         Just userId -> do
```

# Problem 1



Write a function *numnodes* :: *Tree a* -> *Int* that, given a multiway tree, returns the number of nodes.

Examples:

```
numnodes (Node 'a' [Node 'b' []])  
-> 2
```

```
numnodes (Node 'a' [Node 'b' [Node 'c' []]])  
-> 3
```

```
numnodes (Node 'a' [Node 'f' [Node 'g' []], Node 'c' [], Node 'b' [Node  
'd' [], Node 'e' []]])  
-> 7
```

# Haskell Multiway Trees



In Haskell, we define multiway trees as a datatype:

```
data Tree a = Node a [Tree a]
              deriving (Eq, Show)
```

# Example



```
tree1 = Node 'a' [  
    Node 'f' [Node 'g' []],  
    Node 'c' [],  
    Node 'b' [Node 'd' [], Node 'e' []]  
]
```

$$T = 1 + 2 + 1 + 3 = 7$$

