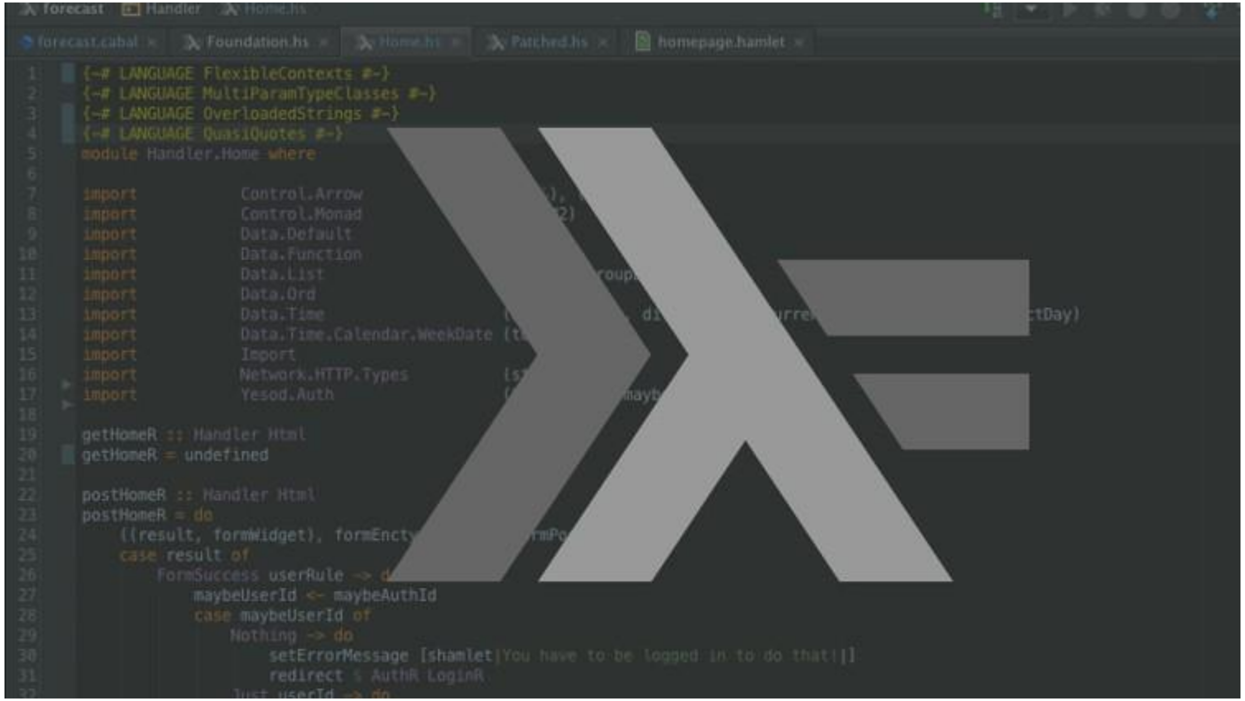
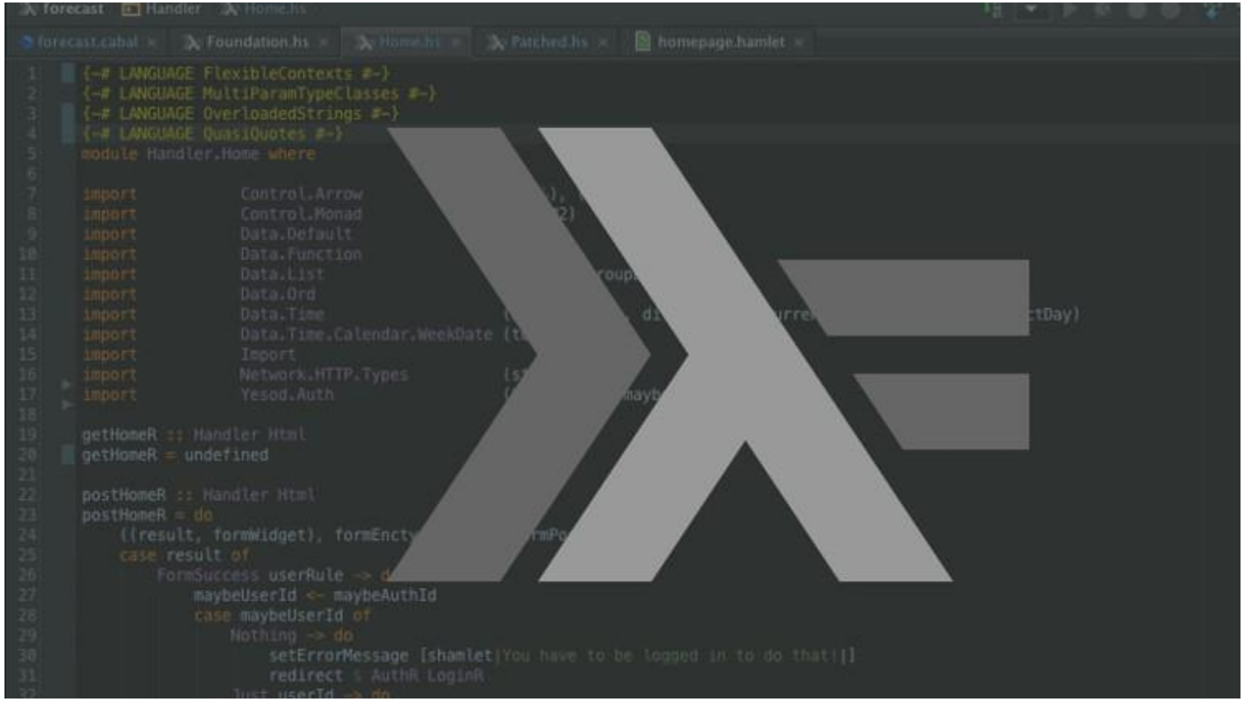
Multiway Tree Problems



```
{-# LANGUAGE FlexibleContexts #-}
{-# LANGUAGE MultiParamTypeClasses #-}
{-# LANGUAGE OverloadedStrings #-}
{-# LANGUAGE QuasiQuotes #-}
module Handler.Home where

import           Control.Arrow
import           Control.Monad
import           Data.Default
import           Data.Function
import           Data.List
import           Data.Ord
import           Data.Time
import           Data.Time.Calendar.WeekDate (t
import           Import
import           Network.HTTP.Types          (s
import           Yesod.Auth

getHomeR :: Handler Html
getHomeR = undefined

postHomeR :: Handler Html
postHomeR = do
    ((result, formWidget), formEncty
    case result of
        FormSuccess userRule -> d
            maybeUserId <- maybeAuthId
            case maybeUserId of
                Nothing -> do
                    setErrorMessage [shamlet|You have to be logged in to do that!|]
                    redirect $ AuthR LoginR
                Just userId -> do
```

Problem 2

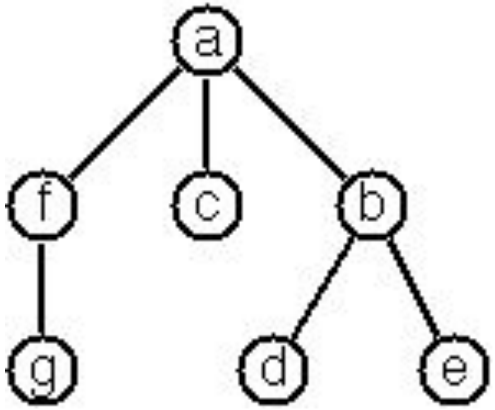


```
{-# LANGUAGE FlexibleContexts #-}
{-# LANGUAGE MultiParamTypeClasses #-}
{-# LANGUAGE OverloadedStrings #-}
{-# LANGUAGE QuasiQuotes #-}
module Handler.Home where

import           Control.Arrow
import           Control.Monad
import           Data.Default
import           Data.Function
import           Data.List
import           Data.Ord
import           Data.Time
import           Data.Time.Calendar.WeekDate (t
import           Import
import           Network.HTTP.Types          (s
import           Yesod.Auth

getHomeR :: Handler Html
getHomeR = undefined

postHomeR :: Handler Html
postHomeR = do
    ((result, formWidget), formEnct
    case result of
        FormSuccess userRule -> d
            maybeUserId <- maybeAuthId
            case maybeUserId of
                Nothing -> do
                    setErrorMessage [shamlet|You have to be logged in to do that!|]
                    redirect $ AuthR LoginR
                Just userId -> do
```

Problem 2

Write a function *stringToTree :: String -> Tree a* that, given a string, it builds the tree. We suppose that the nodes of a multiway tree contain single characters. In the depth-first order sequence of its nodes, a special character ^ has been inserted whenever, during the tree traversal, the move is a backtrack to the previous level.

By this rule, the tree below is represented as: `afg^^c^bd^e^^^`



Problem 2

```
Examples:

stringToTree "afg^^c^bd^e^^^"
-> Node 'a' [Node 'f' [Node 'g' []],Node 'c' [],Node 'b' [Node 'd'
[],Node 'e' []]]

stringToTree "ka^b^c^^"
-> Node 'k' [Node 'a' [],Node 'b' [],Node 'c' []]
```

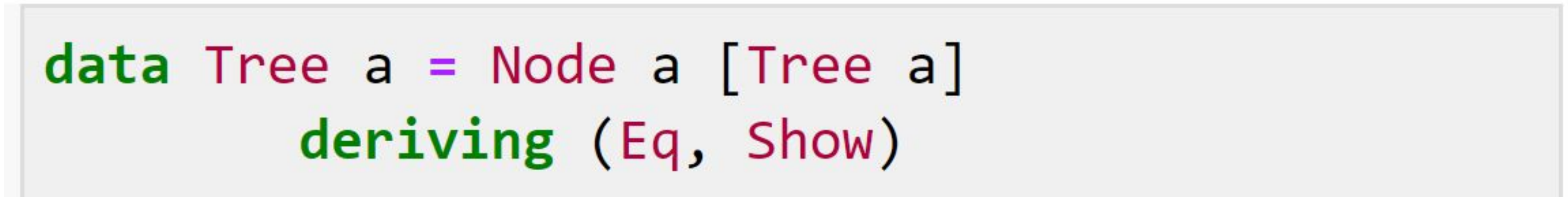
Problem 2

For this problem you will have to return a multiway tree of characters, so use:

stringToTree :: String -> Tree Char

Haskell Multiway Trees

In Haskell, we define multiway trees as a datatype:



```haskell
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' []]
                    ]
```