

Higher Order Functions 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 2

Implement a function *prod* :: *[Int]* -> *Int* that returns the product of a list of integers.

Input

Output

prod [2,10,5] -> 100

prod [3,1,2,4] -> 24

Problem 2

Implement a function *prod* :: *[Int]* -> *Int* that returns the product of a list of integers.

Input

Output

prod [2,10,5] -> 100

prod [3,1,2,4] -> 24

foldl

Type: $(a \rightarrow b \rightarrow a) \rightarrow a \rightarrow [b] \rightarrow a$

Input: `foldl (/) 64 [4,2,4]`

Output: 2.0

Instructor Youtube Channel: Lucas Science

