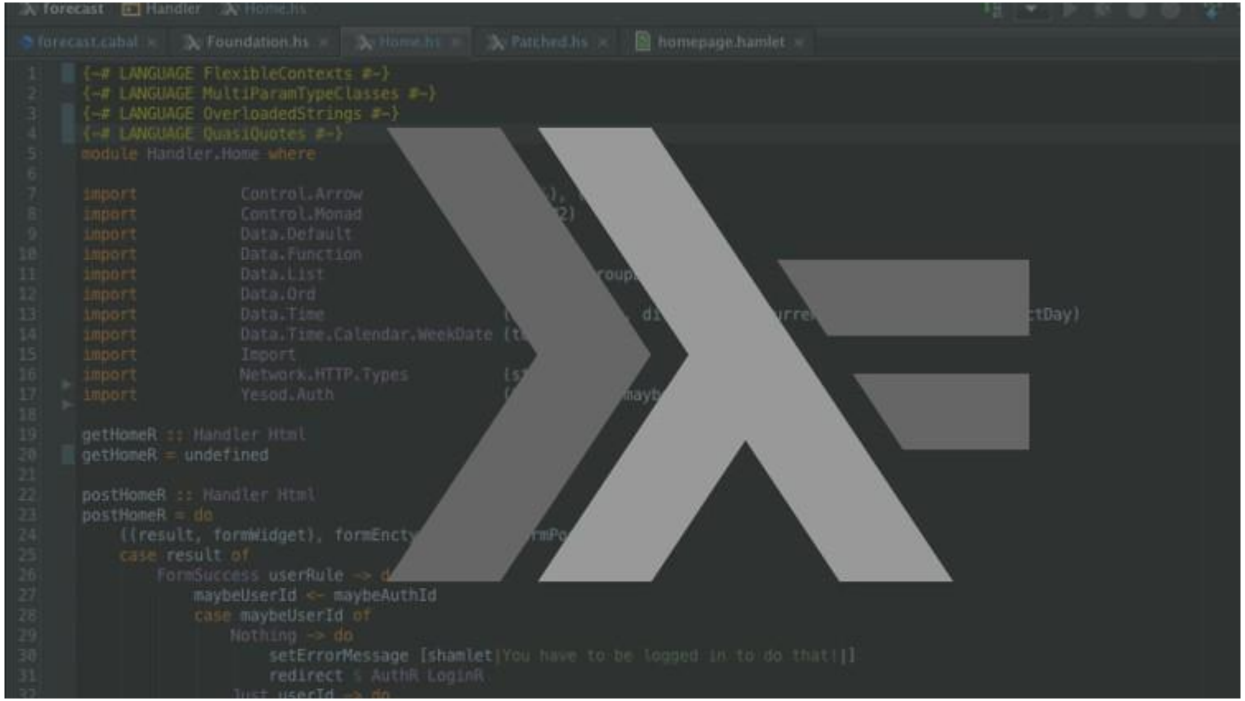
Input and Output Problems



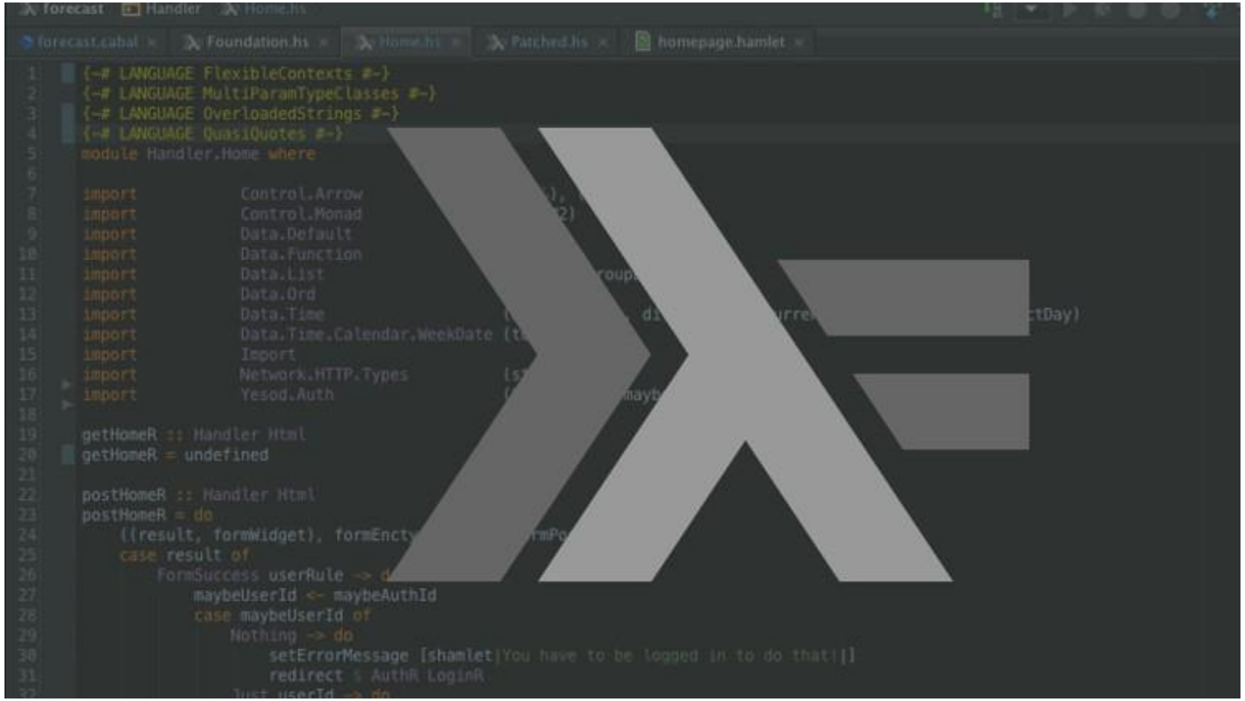
```
{-# LANGUAGE FlexibleContexts #-}
{-# LANGUAGE MultiParamTypeClasses #-}
{-# LANGUAGE OverloadedStrings #-}
{-# LANGUAGE QuasiQuotes #-}
module Handler.Home where

import           Control.Arrow                    ), 
import           Control.Monad                    2)
import           Data.Default
import           Data.Function
import           Data.List                        roup
import           Data.Ord
import           Data.Time                   (         , di           urre                    tDay)
import           Data.Time.Calendar.WeekDate (t
import           Import
import           Network.HTTP.Types          (s
import           Yesod.Auth                  (               mayb

getHomeR :: Handler Html
getHomeR = undefined

postHomeR :: Handler Html
postHomeR = do
    ((result, formWidget), formEncty               rmPo
    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 4



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

import          Control.Arrow                      ), (
import          Control.Monad                      2)
import          Data.Default
import          Data.Function
import          Data.List                          roup
import          Data.Time                 (          , di       urre                        tDay)
import          Data.Time.Calendar.WeekDate (t
import          Import
import          Network.HTTP.Types        (s
import          Yesod.Auth                (              mayb

getHomeR :: Handler Html
getHomeR = undefined

postHomeR :: Handler Html
postHomeR = do
    ((result, formWidget), formEncty              rmPo
    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 4

Write a program which repeatedly reads integers (one per line) until finding a zero value and outputs a sorted version of the inputs read.

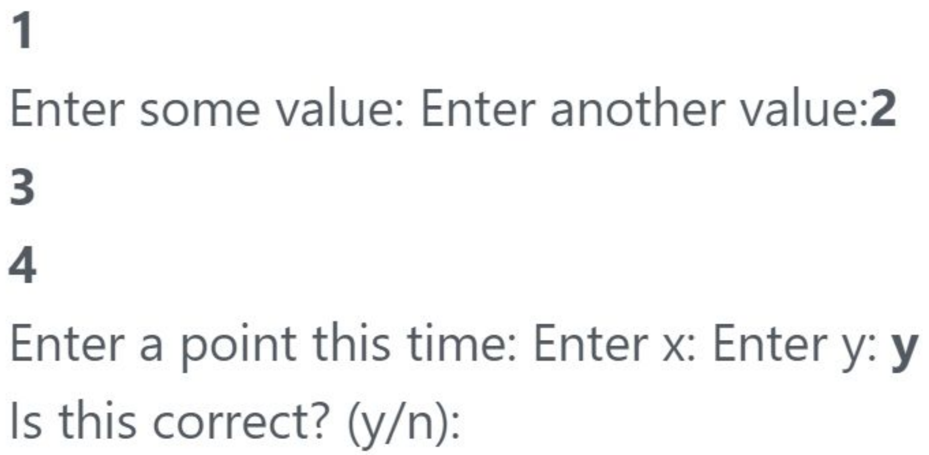
Problem 4

Write a program which repeatedly reads integers (one per line) until finding a zero value and outputs a sorted version of the inputs read.

```
Enter a number (0 to end) 9
Enter a number (0 to end) 4
Enter a number (0 to end) 6
Enter a number (0 to end) 1
Enter a number (0 to end) 5
Enter a number (0 to end) 0
[1,4,5,6,9]
```

Importance of Flushing

Flushing avoids this:



```
1
Enter some value: Enter another value:2
3
4
Enter a point this time: Enter x: Enter y: y
Is this correct? (y/n):
```