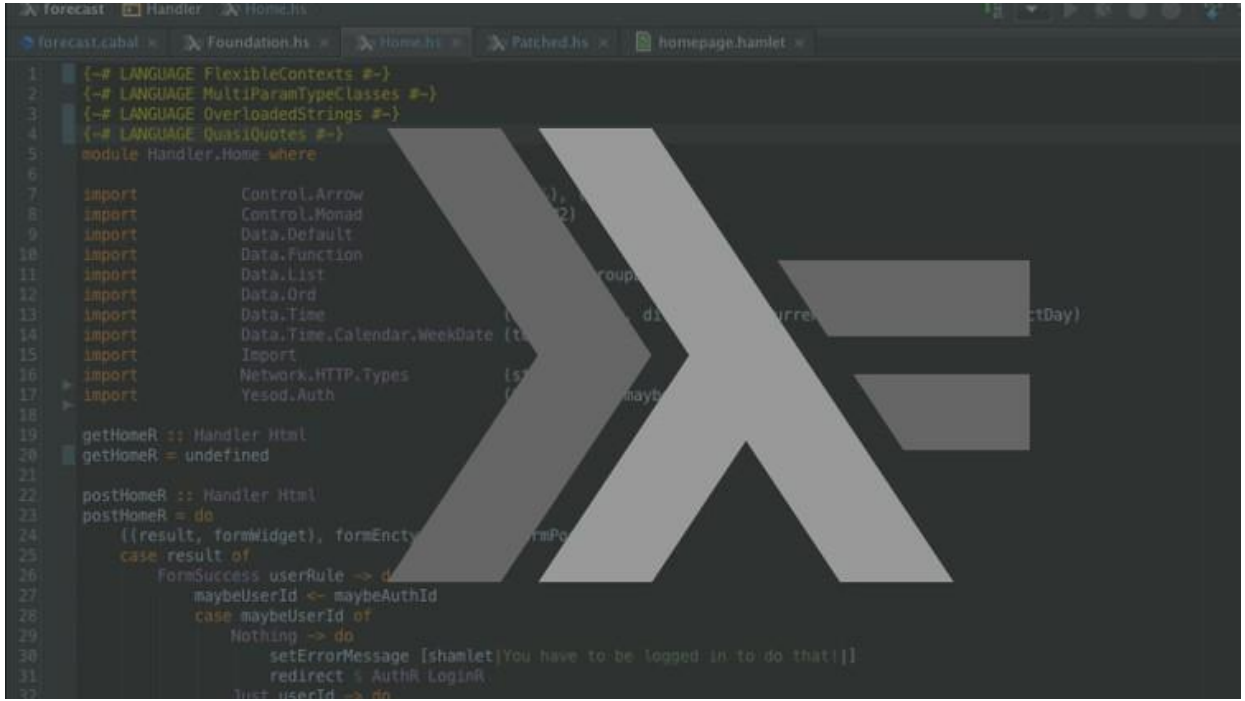
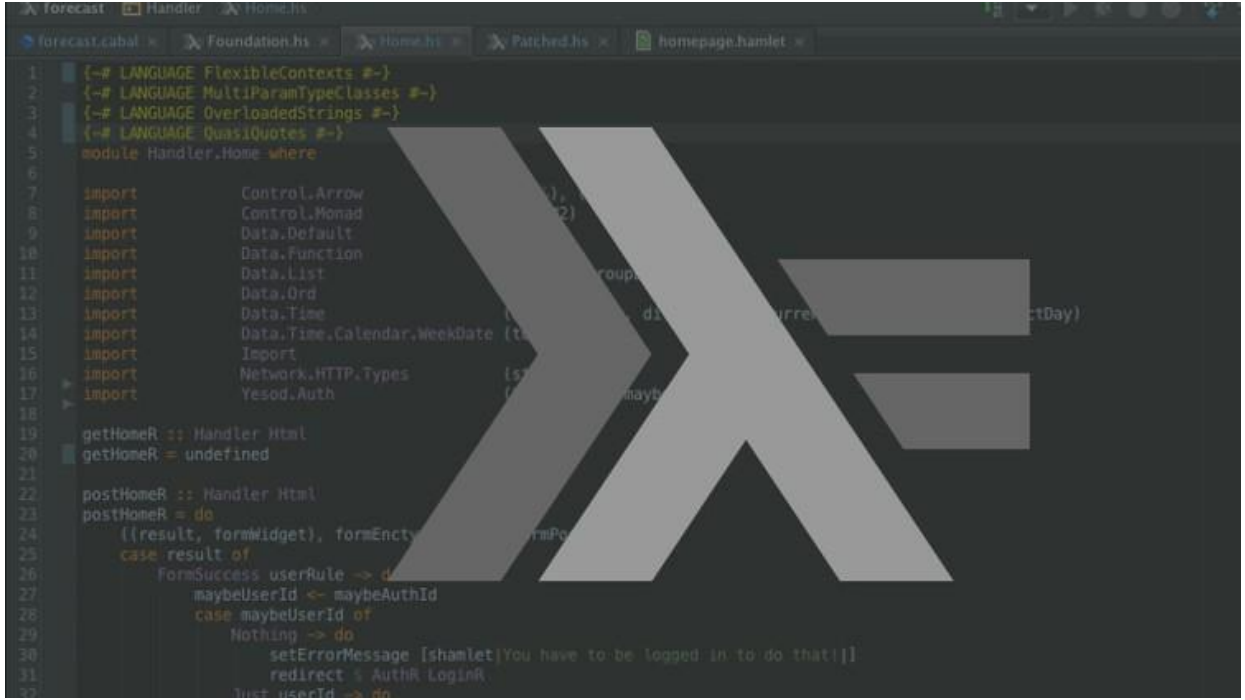
Graph Problems



Problem 2


forecast.cabal
Foundation.hs
Home.hs
Patched.hs
homepage.hamlet
{-# 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
import Import
import Network.HTTP.Types
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

Problem 2

Write a function *depthFirst :: Graph -> Node -> [Node]* that generates a depth-first order graph traversal sequence.

Use the following Graph Notation:

```
type Node = Int
type Edge = (Node,Node)
type Graph = ([Node],[Edge])
```

Problem 2

Write a function *depthFirst :: Graph -> Node -> [Node]* that generates a depth-first order graph traversal sequence.

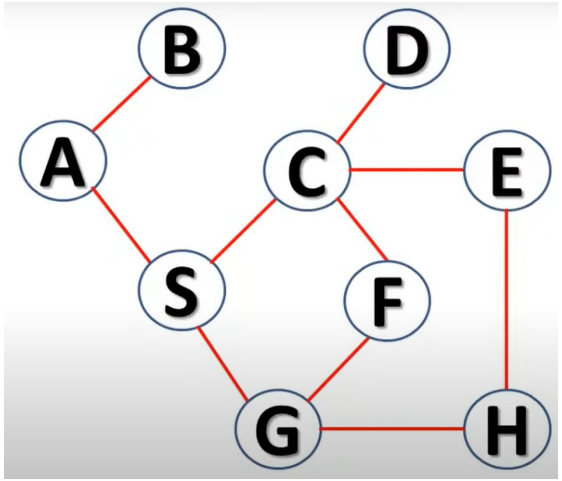
```
Examples

depthFirst
([1,2,3,4,5,6,7],[(1,2),(2,3),(1,4),(3,4),(2,5),(5,4),(6,7)]) 1

-> [1,2,3,4,5]
```

Depth-First Search

The algorithm starts at a starting node and explores as far as possible along each branch before backtracking.

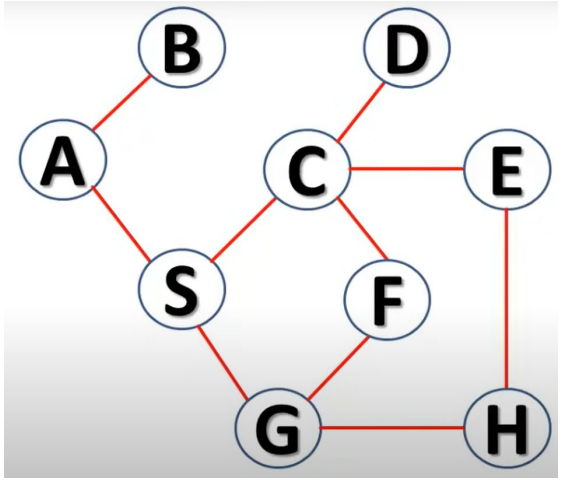


A B S C D E H G F

Depth-First Search

The algorithm starts at a starting node and explores as far as possible along each branch before backtracking.

We start at A



A B S C D E H G F