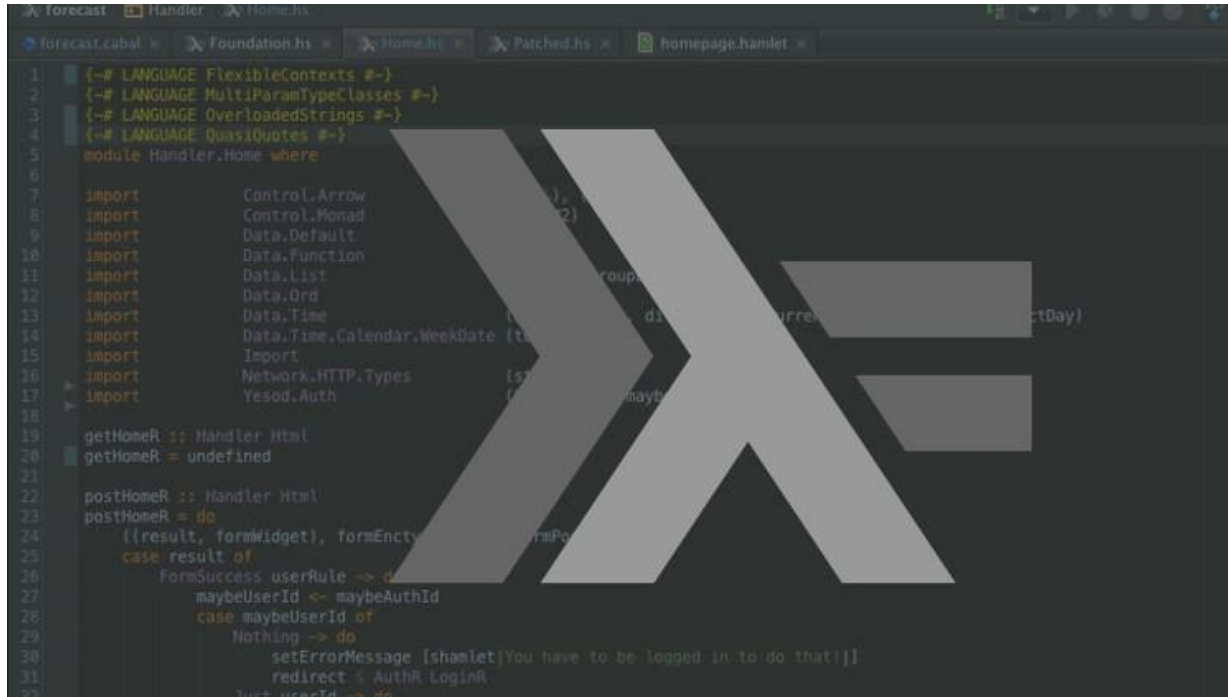


Graph Problems



```
1 {-# LANGUAGE FlexibleContexts #-}
2 {-# LANGUAGE MultiParamTypeClasses #-}
3 {-# LANGUAGE OverloadedStrings #-}
4 {-# LANGUAGE QuasiQuotes #-}
5 module Handler.Home where
6
7 import Control.Monad
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 Network.HTTP.Types
16 import Yesod.Auth
17
18 getHomeR :: Handler Html
19 getHomeR = undefined
20
21 postHomeR :: Handler Html
22 postHomeR = do
23   ((result, formWidget), formEnctype) <- runPost
24   case result of
25     FormSuccess userRule -> do
26       maybeUserId <- maybeAuthId
27       case maybeUserId of
28         Nothing -> do
29           setErrorMessage [shamlet|You have to be logged in to do that!|]
30           redirect % AuthR.LoginR
31         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
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 (toDayOfYear, dayToWeek)
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) <- runPost
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 `acyclicPaths :: Eq a => a -> a -> [(a,a)] -> [[a]]` that given two nodes a and b in a graph, returns all the acyclic paths from a to b.

Examples

```
acyclicPaths 1 4 [(1,2),(2,3),(1,3),(3,4),(4,2),(5,6)]
```

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

```
acyclicPaths 2 6 [(1,2),(2,3),(1,3),(3,4),(4,2),(5,6)]
```

```
-> []
```

Problem 1



Note that the edges have directions.

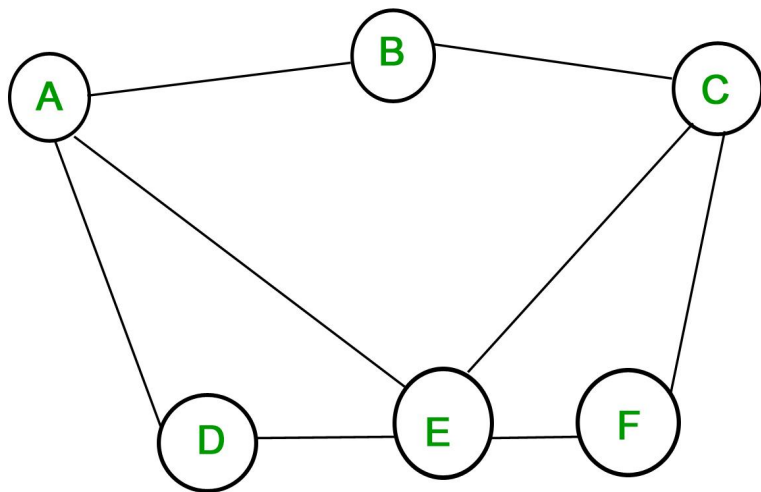
In this problem edge $(1,2)$ is **not** equal to edge $(2,1)$

As one example.

Cycle



In graph theory, a cycle in a graph is a non-empty trail in which only the first and last vertices are equal.



**B-C-E-A-B
is a cycle
example**

Edge Notation

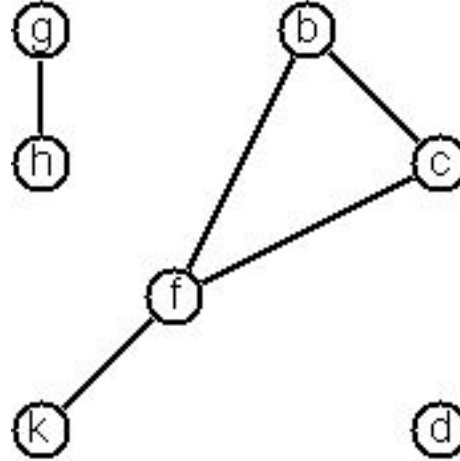


We represent the graph by its edges.

```
data Graph a = Edge [(a, a)]  
               deriving (Show, Eq)
```

Isolated nodes cannot be represented.

Example



Edge $[(g,h), (k,f), (f,b), (f,c), (b,c)]$