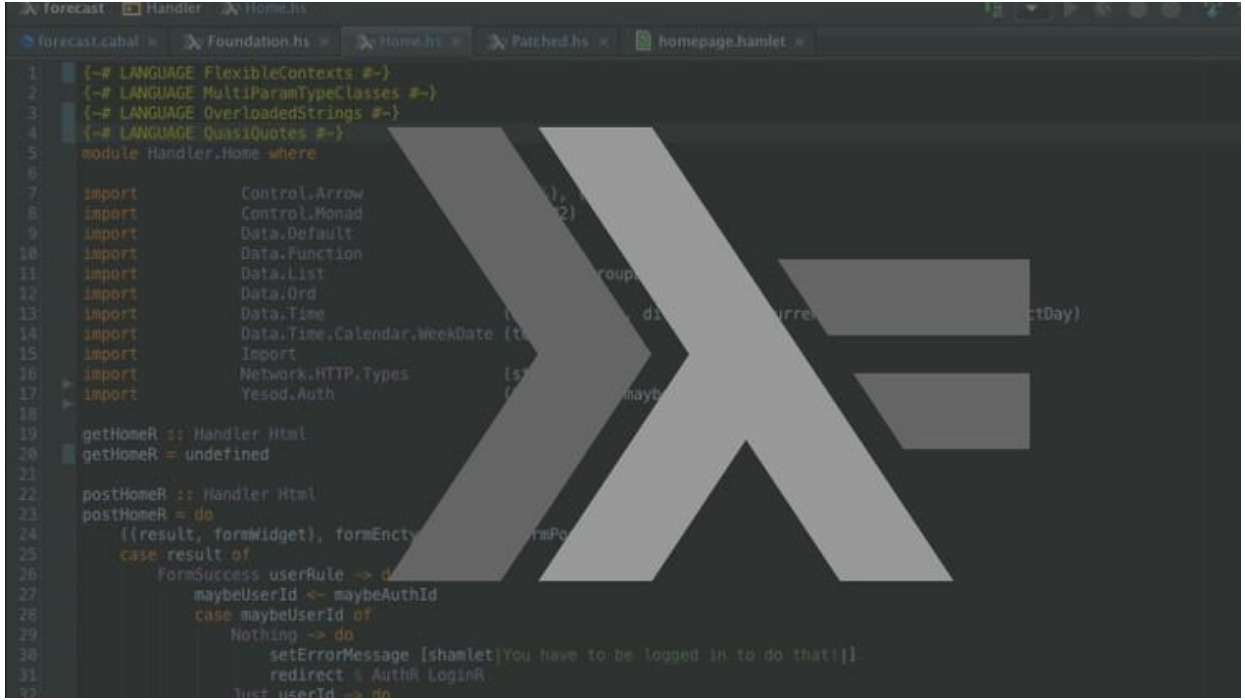
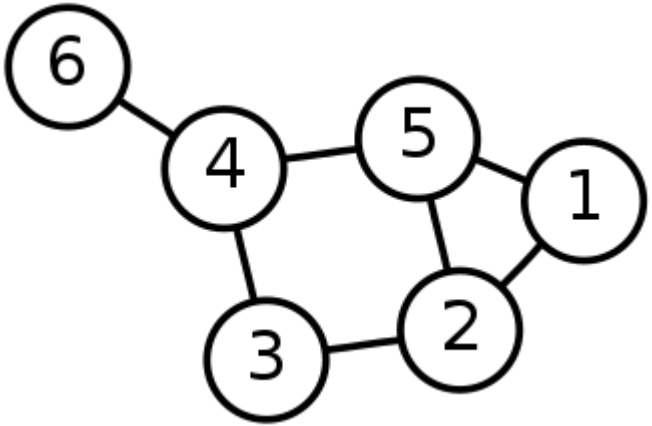
Graphs


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
case result of
FormSuccess userRule
maybeUserId <- maybeAuthId
case maybeUserId of
Nothing -> do
setErrorMessage [shamlet|You have to be logged in to do that!|]
redirect % AuthR LoginR

Graphs

A graph is a structure made of vertices and edges.



A graph with six vertices
and seven edges

Haskell Graphs Notation

In Haskell, we can use several notations for representing Graphs.

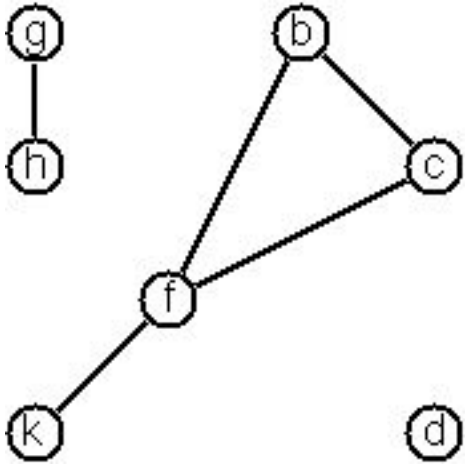
Edge Notation

We represent the graph by its edges.

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

Isolated nodes cannot be represented.

Example



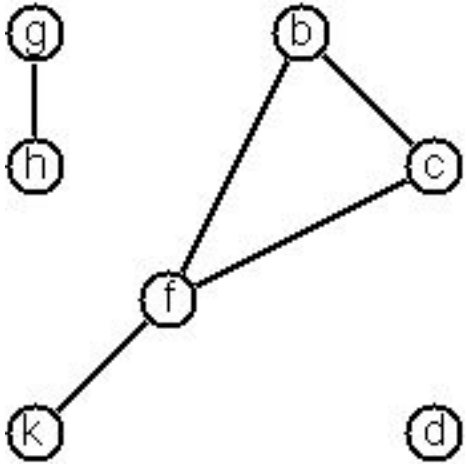
Edge [(g,h), (k,f), (f,b), (f,c), (b,c)]

Graph Notation

We represent the graph by its nodes and its edges.

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

Example



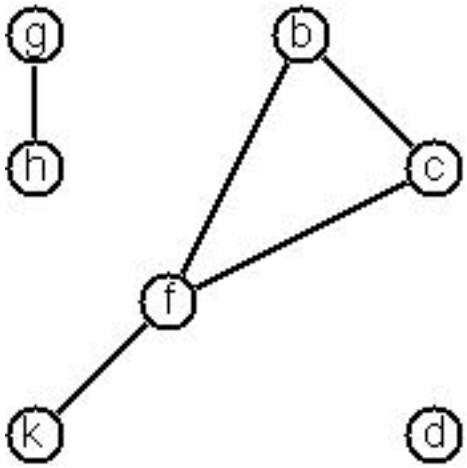

```
Graph ([b,c,d,f,g,h,k],[(b,c),(b,f),(c,f),(f,k),(g,h)])
```

Adjacency List Notation

We represent the graph by its adjacency list.

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

Example




```
Adj   [('b',   "cf"),   ('c',   "bf"),   ('d',   ""),
('f', "bck"), ('g', "h"), ('h', "g"), ('k', "f")]
```

Instructor Youtube Channel: Lucas Science

Lucas Science

SUBSCRIBE