Exercise File 4

Exercise File 4

```
module E4 where

import Data.List
```

Exercise 4.1: Public Announcements

Recall the following functions from the lecture today:

```
(!) :: Eq a => [(a,b)] -> a -> b
(!) v x = let (Just y) = lookup x v in y

(?) :: Eq a => [[a]] -> a -> [a]
(?) lls x = head (filter (x `elem`) lls)

type Prop = Int
type Ag = String
data Form = P Prop | Neg Form | Con Form Form | K Ag Form  deriving (Eq,Ord,Show)

dis :: Form -> Form -> Form
dis f g = Neg (Con (Neg f) (Neg g))

type World = Int
type Relations = [(Ag, [[World]])]
type Valuation = [(World, [Prop])]
data Model = Mo { worlds :: [World]
                , rel :: Relations
                , val :: Valuation }
  deriving (Eq,Ord,Show)

isTrue :: (Model,World) -> Form -> Bool
isTrue (m,w) (P p)     = p `elem` (val m ! w)
isTrue (m,w) (Neg f)   = not (isTrue (m,w) f)
isTrue (m,w) (Con f g) = isTrue (m,w) f && isTrue (m,w) g
isTrue (m,w) (K i f)   = and [ isTrue (m,w') f | w' <- (rel m ! i) ? w ]
```

Implement the `announce` function to make public announcements:

```
announce :: Model -> Form -> Model
announce oldModel f = Mo newWorlds newRel newVal where
  newWorlds = undefined
  newRel    = undefined
  newVal    = undefined
```

Use this to solve the muddy children example:

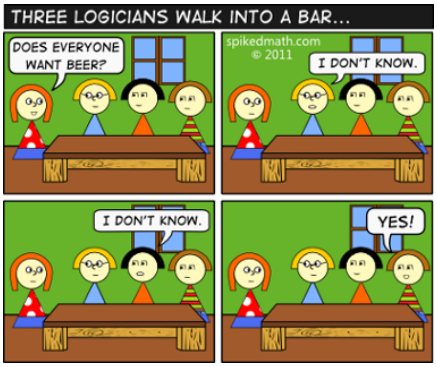
```
muddy :: Model
muddy = Mo
  [0,1,2,3,4,5,6,7]
  [("1",[[0,4],[2,6],[3,7],[1,5]])
  ,("2",[[0,2],[4,6],[5,7],[1,3]])
  ,("3",[[0,1],[4,5],[6,7],[2,3]])]
  [(0,[ ]),(1,[   3]),(2,[   2]),(3,[   2, 3])
  ,(4,[1]),(5,[1, 3]),(6,[1, 2]),(7,[1, 2, 3])]

father :: Form
father = dis (P 1) (dis (P 2) (P 3))

muddy2 :: Model
muddy2 = announce muddy father
```

And so on ...

Exercise 4.2: Drinking Logicians



Note that this is dual to the Muddy Children example. Write a model for the drinking logicians and check:

- After the first logician says "I don't know." it is common knowledge that she wants beer.
- The sequence of all four announcements is possible iff everyone wants beer.
- After the second logician says "I don't know." it is common knowledge that the third logician knows whether everyone wants beer.

Use your model to verify or falsify the following statements:

- After the second logician says "I don't know." it is common knowledge that the third logician knows *that* everyone wants beer.

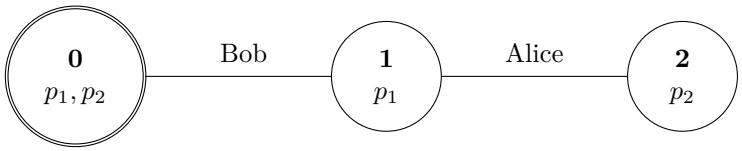
- If the third logician would say *No!* it would be common knowledge that only the first two want beer and the third does not.

Exercise 4.3: Kripke Models and Knowledge Structures

Part i) Find a Kripke model which is equivalent to this knowledege structure:

$$\mathcal{F} = (V = \{p, q, r\}, \theta = (p \vee q) \to r, O_a = \{p\}, O_b = \{q\})$$

Part ii) Find a knowledge structure which is equivalent to this Kripke model:




```
KrM
  [0,1,2]
  [("Alice",[[0],[1,2]]),("Bob",[[0,1],[2]])]
  [(0,[(P 1,True),(P 2,True)])
  ,(1,[(P 1,True),(P 2,False)])
  ,(2,[(P 1,False),(P 2,True)])]
```

Exercise 4.4: Cheryl's Birthday

The following is Question 24 from the Singapore and Asian Schools Math Olympiad 2015. At some point it got "viral", as they say.

> Albert and Bernard just become friends with Cheryl, and they want to know when her birthday is. Cheryl gives them a list of 10 possible dates:
>
> May 15, May 16, May 19, June 17, June 18, July 14, July 16, August 14, August 15, August 17
>
> Cheryl then tells Albert and Bernard separately the month and the day of her birthday respectively. Then the following dialogue takes place.
>
> Albert: I don't know when Cheryl's birthday is, but I know that Bernard does not know too. Bernard: At first I don't know when Cheryl's birthday is, but I know now. Albert: Now I also know when Cheryl's birthday is.
>
> So when is Cheryl's birthday?

You can find a solution of the puzzle using the explicit model checker DEMO-S5 at https://w4eg.de/malvin/illc/cheryl.

Can you solve this puzzle using SMCDEL?

Exercise 4.5: The Dining Cryptographers

Read about the Dining cryptographers problem on Wikipedia, then look at the SMCDEL formalization of it. Extend it to the case with four agents.

Exercise 4.6: Binary Decision Diagrams

Draw binary decision diagrams for the following formulas and statements. First try to do it by hand, then check your results with a BDD package like *HasCacBDD* or *SCMDEL web*.

- $p \vee q$
- $(p \wedge q) \to (p \wedge r)$
- At least one of the propositions p, q and r is true.
- Exactly two of the propositions p, q and r are true.

Practicalities

You can use SMCDEL online: https://w4eg.de/malvin/illc/smcdelweb.

Installing it on your own computer is also possible, but it can be a bit difficult to compile the BDD package HasCacBDD which also has C and C++ code. See https://github.com/jrclogic/SMCDEL or try this:

```
stack unpack smcdel
cd smcdel-1.0.0
stack build
```