CS 210

Lab assignment 4
Jens Blanck
16/4/18

In this lab your task is to experiment with STM in Haskell. I have given a stub of a Haskell program to help start your exploration.

```
import Control.Concurrent
import Control.Concurrent.STM
import Data.List
import Text.Printf

data Item = Water | Food deriving (Eq, Ord, Show)

data Player = Player
  { name :: String
  , gold :: TVar Int
  , inventory :: TVar [Item]
  }

createPlayer n g is = do
  g' <- newTVar g
  is' <- newTVar is
  return $ Player n g' is'

main :: IO ()
main = do
  players <- atomically . sequence $
    zipWith3 createPlayer ["Alice", "Bob"] [5,0] [[Water], [Food]]
  let alice = players !! 0
      bob = players !! 1
  printCurrentState players
  putStrLn "-> Alice gives Water to Bob."
  atomically $ transferItemSTM Water alice bob
  printCurrentState players

printCurrentState ps = mapM_ printPlayer ps

printPlayer p = do
  g <- atomically $ readTVar (gold p)
  is <- atomically $ readTVar (inventory p)
  printf "%s has %d gold and the following items: %s\n" (name p) g (show is)

transferGoldSTM :: Int -> Player -> Player -> STM ()
```

```
transferGoldSTM amount source target = undefined
-- do

transferItemSTM :: Item -> Player -> Player -> STM ()
transferItemSTM item source target = do
  s <- readTVar (inventory source)
  t <- readTVar (inventory target)
  writeTVar (inventory source) (delete item s)
  writeTVar (inventory target) (item : t)

buyItemSTM :: Item -> Int -> Player -> Player -> STM ()
buyItemSTM item amount buyer seller = undefined
-- do
```

1. Implement `transferGoldSTM`, and extend the main method to show that it works as expected.
2. Implement `buyItemSTM`. It should check that the buyer has enough funds and that the seller has the item and otherwise block. Exemplify in main method again. [Hint: Use that STM actions are composable, and use `retry`.]
3. Exemplify in main how to use `orElse` to choose between buying items, where the first fails for either reason.