```
-- Informatics 1 Functional Programming
-- Final Exam #1 - December 2009
--
-- Solutions

import Data.Char


-- Question 1

-- 1a


isVowel x =
  x == 'a' || x == 'e' || x == 'i' || x == 'o' || x == 'u' ||
  x == 'A' || x == 'E' || x == 'I' || x == 'O' || x == 'U'

f :: String -> Bool
f xs = and [ isUpper x | x <- xs, isVowel x]

-- 1b

g :: String -> Bool
g [] = True
g (x:xs) | isVowel x = isUpper x && g xs
         | otherwise = g xs

-- 1c

h :: String -> Bool
h xs = foldr (&&) True (map isUpper (filter isVowel xs))


-- Question 2

-- 2a

p :: [a] -> [a] -> [a]
p xs ys = concat [ [x,y] | (x,y) <- zip xs ys ]
            ++ drop (length xs) ys
            ++ drop (length ys) xs

-- 2b

q :: [a] -> [a] -> [a]
q [] ys = ys
q xs [] = xs
q (x:xs) (y:ys) = x : y : q xs ys
```

```
-- Question 3

type Point = (Int,Int)
data Points = Rectangle Point Point
            | Union Points Points
            | Difference Points Points

-- 3a

inPoints :: Point -> Points -> Bool
inPoints (x,y) (Rectangle (left,top) (right,bottom)) =
    left <= x && x <= right && top <= y && y <= bottom
inPoints p (Union ps qs)      = inPoints p ps || inPoints p qs
inPoints p (Difference ps qs) = inPoints p ps && not (inPoints p qs)


-- 3b

showPoints :: Point -> Points -> [String]
showPoints (a,b) ps = [ makeline y | y <- [0..b] ]
  where
    makeline y =
      [ if inPoints (x,y) ps then '*' else ' ' | x <- [0..a] ]
```