Exercise 18

*-- using newType instead of bool*

newtype All = All { getAll :: Bool }

deriving (Eq, Ord, Read, Show, Bounded)

instance Monoid All where

mempty = All True

(All x) <> (All y) = All (x && y)

*-- define Maybe bind*

define function maybeBind :: Maybe a -> (a -> Maybe b) -> Maybe b

maybeBind Nothing \_ = Nothing

maybeBind (Just x) f = f x

*--define Mayber list*

define function listBind :: [a] -> (a -> [b]) -> [b]

listBind xs f = concat (map f xs)

*-- define Either maybe*

define function eitherBind :: Either r a -> (a -> Either r b) -> Either r b

eitherBind (Left e) \_ = Left e

eitherBind (right x) f = f x

*-- define Arrow bind*

define function arrowBind :: (r -> a) -> (a -> (r -> b)) -> (r -> b)

arrowBind h f r = f (h r) r

*-- define Pairs bind*

define function pairBind :: (r, a) -> (a -> (r, b)) -> (r, b)

pairBind (r1, x) f = (r1 <> r2, y)

where (r2, y) = f x