```
-- Thomas Alessandro Buse 192959, Übung 10, Gruppe 17
-- Paul Rüssmann 196683

import Coalg

import Control.Monad

import Examples

--Aufgabe 1

data STree a = BinS (STree a) a (STree a) | LeftS (STree a) a | RightS a (STree a) | LeafS a deriving
Show


instance Functor STree where

        fmap f (BinS (st1) a (st2)) = BinS (fmap f st1) (f a) (fmap f st2)

        fmap f (LeftS st a) = LeftS (fmap f st) (f a)

        fmap f (RightS a st) = RightS (f a) (fmap f st)

        fmap f (LeafS a) = LeafS (f a)


--Aufgabe 2a)


solutions :: [(Int , Int , Int )]

solutions = [ (x,y,z) | z <- [0..] , y <- [0..z^2] , x <- [0..z^2], 2*x^3 + 5*y + 2 == z^2]

solutions' = do

        z <- [0..]

        y <- [0..z^2]

        x <- [0..z^2]

        guard $ (2*x^3 + 5*y + 2 == z^2)

        return (x,y,z)

--b)


solutions'' = [0..] >>= (\z -> [0..z^2] >>= (\y -> [0..z^2] >>= (\x -> if 2*x^3 + 5*y + 2 == z^2 then  [(x,y,z)]
else  [] )))
```

```
--Aufgabe 3


preorderM :: MonadPlus m => Bintree a -> m a

preorderM Empty = mzero

preorderM (Fork a (l) (r)) = (return a) `mplus` preorderM (l) `mplus` preorderM (r)




--Aufgabe 4

sdiv :: Int -> Int -> Maybe Int

_ `sdiv` 0 = Nothing

x `sdiv` y = Just $ x `div` y



f x y z = do

        a <- 18 `sdiv` x

        b <-  a `sdiv` y

        c <-  6 `sdiv` z

        return $ b+c
```