Polymorphism/Polymorphism Restrictions

1 Polymorphism

1.1 Polymorphic function

```
length :: [a] -> Int
length [] = 0
length (x:xs) = 1 + length xs
```

1.2 Restrictions

- These functions have a restriction which forces them to only return their parameters in special way to follow its type signature.
- These functions and their type signatures only allow for one implementation in code

```
id :: a -> a
id x = x

aToB :: a -> b
aToB x = undefined
aToB' :: a
aToB' = undefined
aToB'' :: a -> b -> c -> ...
aToB'' x y ... = undefined

const :: a -> b -> a
const x _ = x

seq :: a -> b -> b
seq _ y = y

hasTypeOf :: a -> a -> a
hasTypeOf x y = x

uncurry :: (a -> b -> c) -> (a,b) -> c
uncurry f (x,y) = f x y

curry :: ((a,b) -> c) -> a -> b -> c
curry f x y = f (x,y)

flip :: (b -> a -> c) -> a -> b -> c
flip f x y = f y x
```

```
f1 :: a -> [a]
f1 x = replicate 5 x

-- Only possibility
f3 :: a -> [b]
f3 x = []

-- Not as restrictive since "a" is a number,
-- therefore we can map b to many more possibilities
f4 :: (Num a) => a -> [b]
f4 x = []
```