FUNCTIONAL PROGRAMMING 2018

MOISE GABRIEL

QUESTION 1

```
> data Tree a = One a | Two (Tree a) (Tree a)
```

(b)

```
> size :: Tree a -> Int
> size (One _) = 1
> size (Two l r) = 1 + size l + size r
```

(c)

```
> foldTree :: (a -> b) -> (b -> b -> b) -> Tree a -> b
> foldTree one two (One x) = one x
> foldTree one two (Two x y) = two (foldTree one two x) (foldTree one two y)
```

(a) Tree is the new datatype and One and Two are its datatype constructors.

---

foldTree One Two = id, so this is the natural fold for the Tree datatype

(d)

```
> size' :: Tree a -> Int
> size' = foldTree (const 1) (\ x y -> 1+x+y)
```

(e)

```
> subs :: Tree a -> [Tree a]
> subs (One x) = [One x]
> subs (Two x y) = (Two x y) : subs x ++ subs y
```

(f)

```
> subs' :: Tree a -> [Tree a]
> subs' = foldTree (\x -> [One x]) (\ lt rt -> (Two (head lt) (head rt)) : lt ++ rt)
```

(g)

```
> proper :: Tree a -> [Tree a]
> proper (One _) = []
> proper (Two (One x) y) = (One x) : y : proper y
> proper (Two x (One y)) = x : proper x ++ [One y]
> proper (Two l r) = proper l ++ proper r
```

(h) Let's suppose that proper = foldTree f g

Then, we have

proper (One _) = [] ⇒ f _ = [] ⇒ f = const []

proper (Two (One 1) (One 2)) = [One 1, One 2]

foldTree (const []) g (Two (One 1) (One 2)) = g (foldTree (const []) g (One 1)) (foldTree (const []) g (One 2)) = g [] [] ⇒ g [] [] = [One 1, One 2]

Same goes for foldTree (Two (One 3) (One 4)) = g [] [] ⇒ g [] [] = [One 3, One 4] ⇒

⇒ this gets us to a contradiction, so proper cannot be expressed as a fold.