Декларативное программирование

Свертки

Упражнение

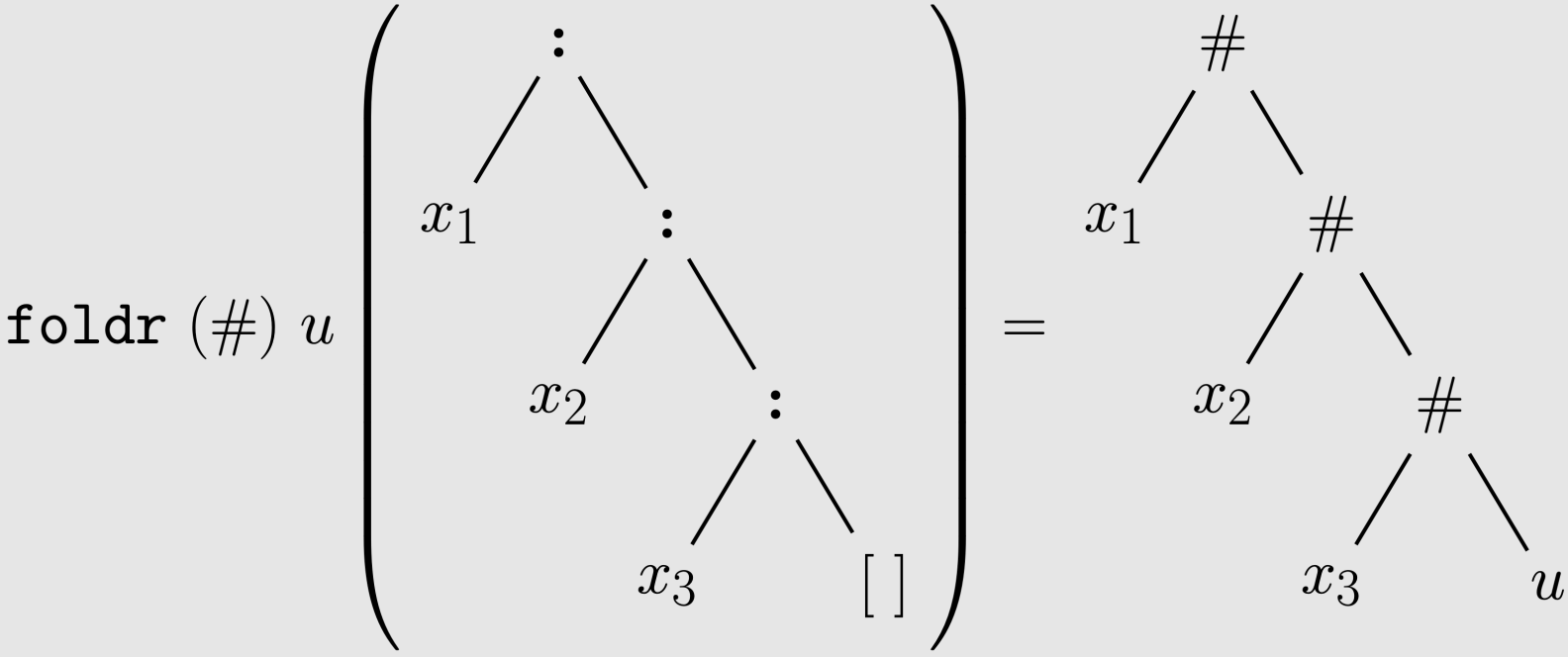
Напишите три варианта функции doubleAll, которая удваивает каждое значение в списке

1. Используя рекурсию
2. Используя list comprehension
3. Используя map

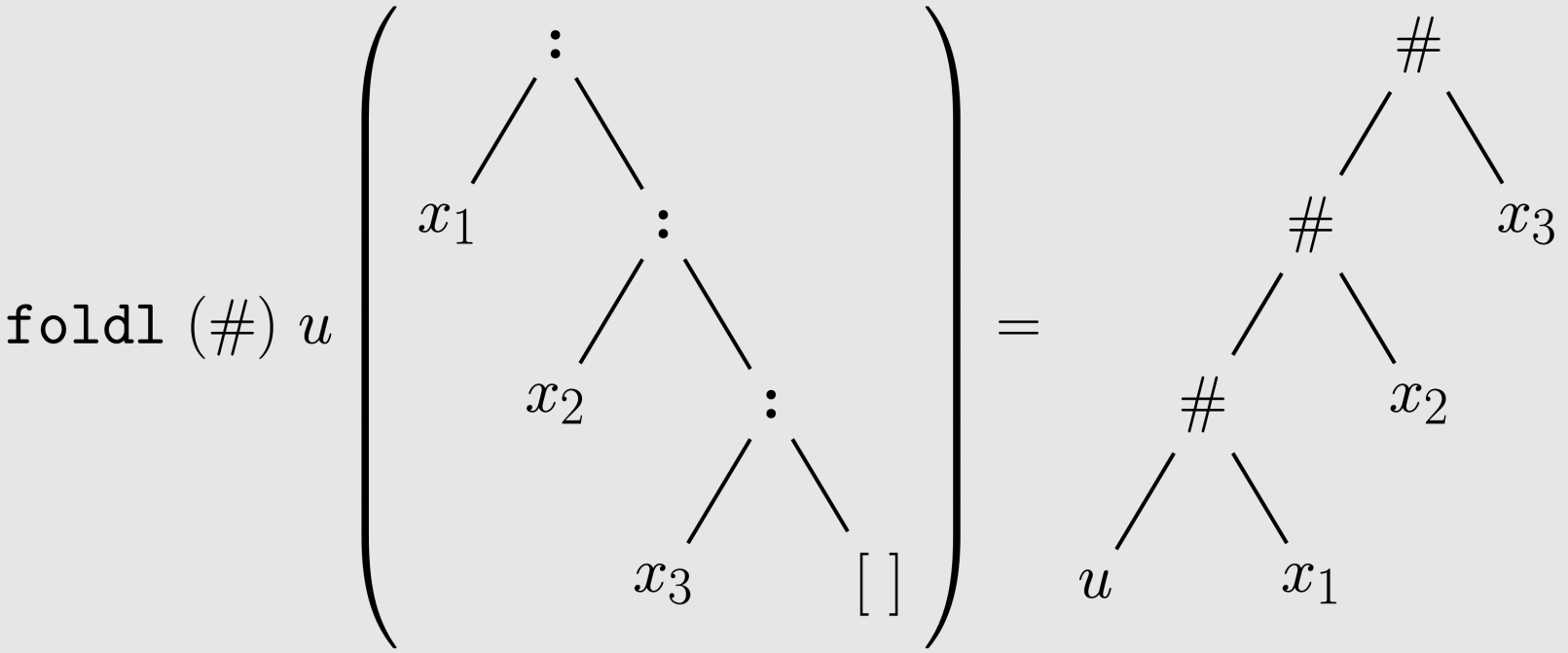
Упражнение

Реализуйте функцию length, используя map и sum

foldr



foldl



Упражнение

- Реализуйте свою версию функции reverse, используя свертку
- Реализуйте свою версию функции last с помощью foldl1

Упражнение

- Реализуйте функцию prefixes, используя свертку:

```
ghci> prefixes "abc"
["", "a", "ab","abc"]
```

- Реализуйте функцию suffixes, используя свертку:

```
ghci> suffixes "abc"
["abc", "bc", "c", ""]
```

Упражнение

Используя свертку (правую или левую?) напишите функцию asInt:

```
asInt :: String -> Int

asInt = undefined


ghci>asInt "123"

123
```

any

```
any' p xs = foldr
            ???
            ???
            xs
any'' p xs = foldl
             ???
             ???
             xs
```

Свертки и бесконечные списки

```
ghci>any' even [1..]
True
ghci>any'' even [1..]
ghci> any' even (repeat 1)
ghci> any'' even (repeat 1)
```

Свертки и бесконечные списки

```
foldl' :: (b -> a -> b) -> b -> [a] -> b
foldl' op acc [] = acc
foldl' op acc (x:xs) = foldl' op (acc `op` x) xs

foldr' :: (a -> b -> b) -> b -> [a] -> b
foldr' op acc [] = acc
foldr' op acc (x:xs) = x `op` foldr' op acc xs
```

undefined

```
ghci> foldr (+) 0 [1, 2, 3, 4, undefined]
*** Exception: Prelude.undefined
ghci> foldr (+) 0 (take 4 [1, 2, 3, 4, undefined])
10
ghci> length [1, 2, 3, 4, undefined]
5
ghci> length ([1, 2, 3, 4] ++ undefined)
*** Exception: Prelude.undefined
ghci> let xs = [1, 2] ++ undefined
ghci> length $ take 2 $ take 4 xs
2
```

undefined

```
ghci> foldr (\_ _ -> 2023) 0 [1..5]
2023
ghci> foldr (\_ _ -> 2023) 0 [1, 2, 3, undefined]
2023
ghci> foldr (\_ _ -> 2023) 0 ([1, 2, 3] ++ undefined)
2023
ghci> foldr (\_ _ -> 2023) 0 undefined
*** Exception: Prelude.undefined
ghci> foldr (\_ _ -> 2023) 0 [undefined]
2023
```

scanl, scanr

```
foldr :: (a -> b -> b) -> b -> [a] -> b
scanr :: (a -> b -> b) -> b -> [a] -> [b]

foldl :: (b -> a -> b) -> b -> [a] -> b
scanl :: (b -> a -> b) -> b -> [a] -> [b]
```

scanl

```
scanl' :: (b -> a -> b) -> b -> [a] -> [b]
scanl' f ini [] = [ini]
scanl' f ini (x:xs) = ini : scanl' f (ini `f` x) xs
```

Упражнение

- Реализуйте функцию prefixes, используя scanl или scanr:

```
ghci> prefixes "abc"
["", "a", "ab","abc"]
```

- Реализуйте функцию suffixes, используя scanl или scanr :

```
ghci> suffixes "abc"
["abc", "bc", "c", ""]
```

scanl и бесконечные списки

```
factorials :: (Num a, Enum a) => [a]
factorials = scanl (*) 1 [1..]

partialSums :: Num a => [a] -> [a]
partialSums = scanl (+) 0
```

Упражнение

Используя scanl, создайте бесконечный список чисел Фибоначчи

```
fibs :: [Integer]
fibs = undefined
```

Упражнение (e)

Используя foldl, foldr, scanl или scanr создайте список следующего вида:

$$sae\ n = \left[\sum_{i=0}^{i=1} \frac{1}{i!}, \sum_{i=0}^{i=2} \frac{1}{i!}, \sum_{i=0}^{i=3} \frac{1}{i!}, \ldots, \sum_{i=0}^{i=n} \frac{1}{i!}\right].$$