

Летучка 7 · 7 минут · без конспектов · Ваше ФИ: _____

1. Кайнд конструктора типа `Maybe`: _____
2. Один шаг β -редукции: $(\lambda x. x) a \rightsquigarrow$ _____
3. Результат `foldr (-) 0 [1, 2]`: _____
4. Нейтральный элемент для `(++)` на списках: _____
5. Дополните кодирование списка Чёрча `[1, 2]`: $\lambda s z.$ _____

✂ _____

Листок 7. Свёртки

Задание 1 · слайд 4. Напишите полные инстансы `Semigroup` и `Monoid` для списка в обёртке `Batch`. Объединение сохраняет порядок всех элементов.

```
newtype Batch a = Batch [a]
```

Задание 2 · слайд 5. Напишите полные инстансы `Semigroup` и `Monoid` для статистики тестов. При объединении блоков складываются число ошибок и общее число тестов.

```
data Counts = Counts { errors :: Int, total :: Int }
```

Задание 3 · слайд 6. Напишите полные инстансы `Semigroup` и `Monoid` для своего варианта `Maybe`. `Missing` означает отсутствие результата; два `Present` объединяют содержащиеся значения. Выберите необходимые ограничения на `a`.

```
data Optional a = Missing | Present a
```

Задание 4 · слайд 7. Напишите `stats :: [Bool] -> Counts`, используя готовый инстанс: `True` — успешный тест, `False` — ошибка. Для `[True, False, True]` получите `Counts 1 3`, для пустого списка — `Counts 0 0`. Обойдитесь без явной рекурсии; пригодятся `map` и `mconcat`.

```
mconcat :: Monoid m => [m] -> m
```

Задание 5 · слайд 9. Расставьте скобки и вычислите `foldr (-) 0 [1,2,3]` и `foldl (-) 0 [1,2,3]`.

Задание 6 · слайд 13. Реализуйте скалярное произведение через `zipWith` и `foldr`. При разной длине списков используйте общий префикс.

```
prod :: Num a => [a] -> [a] -> a
```

Задание 7 · слайд 18. Завершите `foldMapList` через `foldr`. Шаг объединяет `f x` с результатом свёртки хвоста.

```
foldMapList :: Monoid m => (a -> m) -> [a] -> m  
foldMapList f = foldr (\x rest -> _____) _____
```

Задание 8 · слайд 20. Напишите полный инстанс `Foldable Tree`: заголовок и метод `foldMap`. Порядок — корень, левое поддерево целиком, затем правое; `Leaf` пуст. Не используйте `deriving Foldable`.

```
data Tree a = Leaf | Node a (Tree a) (Tree a)
```

Задание 9 · слайд 34. Укажите результат, «ошибка» или «не завершится». `xs = True : False : undefined`, `foldl'` — из `Data.List`.

```
foldr (&&) True xs  
foldl' (&&) True xs  
foldr (\x rest -> x == 4 || rest) False [1..]
```

Задание 10 · слайд 35 (факультативно). Напишите полные инстансы `Semigroup` и `Monoid` для `Func`: функции получают один аргумент и объединяют результаты. Выберите ограничения на `a` и `b`.

```
newtype Func a b = Func { runFunc :: a -> b }
```