Экзамен

Задание 1.1 (Racket). Реализуйте функцию `stream-chunks`, которая разбивает входной поток на поток списков заданной длины:

```
(for/list ([i 3]
           [chunk (stream-chunks (in-naturals 1) 4)])
  chunk)

; '((1 2 3 4) (5 6 7 8) (9 10 11 12))
```

Задание 1.2 (Racket). Реализуйте функцию `stream-splits`, строящую поток пар префикс-суффикс так, что конкатенация префикса и суффикса в каждой паре даёт исходный список, и пары упорядочены по возрастанию длины префикса:

```
(stream->list (stream-splits '(a b c)))
; '((() . (a b c))
;   ((a) . (b c))
;   ((a b) . (c))
;   ((a b c) . ()))
```

Задание 1.3 (Haskell). Рассмотрите следующие определения:

```
newtype Parse a = Failed String | Parsed a deriving (Show)
data NonEmpty a = NonEmpty a [a] deriving (Show)

pInt :: String -> Parse Int
```

Используя эти определения, реализуйте функцию `pSomeInts`, которая разбирает последовательность из одного или более целых чисел:

```
>>> pSomeInts ["123", "345", "0"]
Parsed (NonEmpty 123 [345, 0])
```

Задание 1.4 (miniKanren). Реализуйте отношение `replicate°`, такое что `(replicate° n one many)` верно когда список `many` является конкатенацией `n` копий списка `one`:

```
(run* (q) (replicate° (build-num 3) '(a b c) q))
; '((a b c a b c a b c))

(run* (q) (replicate° (build-num 3) q '(a b c a b c a b c)))
; '((a b c))

(run* (q) (replicate° q '(a b c) '(a b c a b c a b c)))
; '((1 1))
```

Задание 1.5 (miniKanren). Реализуйте отношение `words°`, такое что `(words° sentence words)` верно, когда список `sentence` является конкатенацией элементов `words` (возможно, с повторениями), и каждый элемент списка `words` — непустой список:

```
(run* (q) (words° '(a h a t h e h a d) '((h e) (h a d) (a) (h a t))))
; '(_.0)

(run* (q) (words° '(a h a t h e h a d) '((h e) (h a d) (h a t))))
; '()
```