Задание №14
Бесконечные списки

1. Общая постановка задачи

Загрузите с портала пример, который является заготовкой к выполнению задания. Прежде чем выполнять задание внимательно ознакомьтесь с текстом примера.

Текст заготовки разделен на блоки, которые разграничены комментариями.

В блоке 2 примера опишите функции `drop` и `take`, где каждая функция принимает в качестве параметров целое число `n` и поток (бесконечный список). Функция `drop` должна выдавать поток без первых `n` элементов. Функция `take` должна выдавать список из первых `n` элементов потока. В случае недостаточного количества элементов в потоке необходимо удалять/возвращать то количество элементов, которое есть в наличии.

В блоке 3 опишите последовательность (поток) для вашего варианта задания 7. Поток должен определяться в том же порядке, в котором он определялся при решении задания 7.

Поток может быть определен как результат функции, если для его задания необходимы параметры. Если для определения потока параметров не требуется, то он должен быть сохранен в глобальной переменной. Если для выполнения задания потребуются вспомогательные функции, обрабатывающие потоки, то эти функции опишите в блоке 2.

В блоке 4 задайте набор тестовых вызовов, демонстрирующих выполнение вашего задания. Тестовые вызовы должны использовать реализованные функции `take` и `drop`.

Опишите программу в текстовом файле с именем `task14-NN.lsp`, где `NN` — номер вашего варианта задания 7. Полученный файл загрузите на портал в качестве выполненного задания.

2. Предварительные замечания

Поток может быть определен только как результат выполнения одной операции или набора операций над потоками (возможно, с привлечением конечных последовательностей).

Поток не должен определяться в виде поэлементного конструктора с помощью функции независимого вычисления элемента.

При выполнении задания можно использовать нехвостовую рекурсию. Средства LISP для организации циклов использовать нельзя (за исключением рекурсии).

Не следует делать предположений на счет задания, не сформулированных явно в условии. Если возникают сомнения — задайте вопрос на форуме «Язык Lisp».

3. Пример выполнения задания

ЗАДАНИЕ: Описать бесконечную последовательность простых чисел.

РЕШЕНИЕ:
Содержимое файла `task14-NN.lsp`:

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; 1. блок для реализации отложенных вычислений
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; макрос, создающий обещание (thunk из выражения expr)
(defmacro delay (expr)
  `(cons NIL (lambda () ,expr)))

;; функция вычисления обещания
(defun force (delay)
  (if (car delay)
      (cdr delay)
      (progn (setf (car delay) T)
             (setf (cdr delay) (funcall (cdr delay))))))
```

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; 2. блок для реализации потоков
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; макрос для создания потока из головы a и хвоста str
;; результат вычисления str - поток
(defmacro cons-stream (a str)
  `(cons ,a (delay ,str)))

;; функция извлечения головы потока (вместо car-stream)
(defun head (stream)
  (car stream))

;; функция извлечения хвоста потока (вместо cdr-stream)
(defun tail (stream)
  (force (cdr stream)))

;; функция проверки потока на пустоту
(defun empty-stream-p (stream)
  (null stream))

;; определение глобальной переменной - пустой поток
(defvar *the-empty-stream* '())

;; функция фильтрации потока stream с помощью предиката p
(defun filter-stream (p stream)
  (cond ((empty-stream-p stream) *the-empty-stream*)
        ((funcall p (head stream))
         (cons-stream (head stream)
                      (filter-stream p (tail stream))))
        (T (filter-stream p (tail stream)))))

;; ниже должны быть определены вспомогательные функции
;; для работы с потоками для индивидуального задания
;; (defun drop (n stream))
;; (defun take (n stream))
;; возможны еще дополнительные функции в зависимости от задания

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; 3. примеры потоков
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;; глобальная переменная, содержащая поток единиц
(defvar *ones*) ; объявление переменной
(setq *ones* (cons-stream 1 *ones*))

;; функция, выдающая поток целых чисел, начинающийся с n
(defun integers-starting-from (n)
  (cons-stream n (integers-starting-from (+ n 1))))

;; вспомогательная функция, проверяющая делится ли x на y
(defun divisible-p (x y)
  (= (mod x y) 0))

;; функция, получающая поток простых чисел из потока stream при условии,
;; что stream - возрастающая последовательность целых чисел в которой
;; головной элемент - простое число и в stream нет чисел, у которых
;; делителями являются числа меньшие головного элемента (кроме единицы).
(defun sieve (stream)
  (cons-stream
    (head stream)
    (sieve
```

```
      (filter-stream
        (lambda (x) (not (divisible-p x (head stream))))
        (tail stream)))))

;; глобальная переменная, содержащая поток простых чисел
(defvar *primes* (sieve (integers-starting-from 2)))

;; ниже должны быть определены вспомогательные функции/глобальные переменные
;; для индивидуального задания


;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; 4. тестовые примеры
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
(print (head (tail (tail (tail (tail (tail *primes*)))))) ; 6-е простое число
(print
  (head (tail (tail (tail (tail (tail (tail (tail (tail *primes*)))))))) ; 8-е
     число

;; ниже должны быть определены примеры потока
;; для индивидуального задания с использованием функций drop и take
```

Файл с примером можно загрузить с портала.