Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н. Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №5

по курсу «Функциональное и логическое программирование»

Студент Рунов К.А.

Группа ИУ7-64Б

Оценка (баллы)

Преподаватели Толпинская Н. Б., Строганов Ю. В.

2024 г.

```
; 1. Напишите функцию, которая уменьшает на 10 все числа из списка
   -аргумента этой функции, проходя по верхнему уровню списковых я
   чеек. (Список смешанный структурированный)
(defun f1-helper (x)
  (cond ((numberp x) (- x 10))
        (t x)))

(defun f1 (lst)
  (mapcar #'f1-helper lst))
```

```
; 2. Написать фукнцию которая получает как аргумент список чисел,
   а возвращает список квадратов этих чисел в том же порядке
(defun f2-helper (x)
  (cond ((numberp x) (* x x))
        (t x)))

(defun f2 (lst)
  (mapcar #'f2-helper lst))
```

```
; 3. Напишите фукнцию, которая умножает на заданное число
   -аргумент все числа из заданного списка-аргумента, когда
; а) все элементы списка -- числа,
; б) элементы списка -- любый объекты.
(defun f3-helper (x c)
  (cond ((numberp x) (* x c))
        (t x)))

(defun f3 (lst c)
  (mapcar #'(lambda (x) (f3-helper x c)) lst))
```

```
; 4. Написать функцию, которая по своему списку-аргументу lst опре
   деляет является ли он палиндромом (то есть равны ли lst и
   (reverse lst)), для одноуровневого смешанного списка
(defun f4 (lst)
  (equal lst (reverse lst)))
```

```
; 5. Используя функционалы, написать предикат set-equal, который в
   озвращает t, если два его множества-аргумента (одноуровневые сп
   иски) содержат одни и те же элементы, порядок которых не имеет
   значения
```

```
(defun set-equal-helper (lst1 lst2 flag-call-first-time)
  (cond ((and (null lst1) flag-call-first-time) nil)
        ((null lst1) t)
        ((member (car lst1) lst2)
         (set-equal-helper (cdr lst1) lst2 nil))
        (t nil)))

(defun set-equal (lst1 lst2)
  (and (set-equal-helper lst1 lst2 t)
       (set-equal-helper lst2 lst1 t)))
```

```
; 6. Напишите функцию, select-between, которая из списка
   -аргумента, содержащего только числа, выбирает только те, котор
   ые расположены между двумя указанными числами - границами
   -аргументами и возвращает их в виде списка (упорядоченного по в
   озрастанию (+2 балла))
(defun push-ordered (lst elem)
  (cond ((null lst) (list elem))
        ((< elem (car lst)) (cons elem lst))
        (t (cons (car lst) (push-ordered (cdr lst) elem)))))

(defun select-between-helper (lst start end result-lst)
  (cond ((null lst) result-lst)
        ((and (< start (car lst)) (> end (car lst)))
         (select-between-helper (cdr lst) start end
                                (push-ordered result-lst (car lst))))
        (t (select-between-helper (cdr lst) start end
           result-lst))))

(defun select-between (lst start end)
  (select-between-helper lst start end nil))
```

```
; 7. Написать функцию, вычисляющую декартово произведение двух сво
   их списков-аргументов (напомним, что A x B это множество всевоз
   можных пар (a b), где a принадлежит A, b принадлежит B)
(defun decart (lstA lstB)
  (mapcan #'(lambda (a)
              (mapcar #'(lambda (b)
                          (list a b)) lstB)) lstA))
```

```
; 8. Почему так реализовано reduce, в чем причина?
; (reduce #'+ ()) -> 0
```

```
; (reduce #'* ()) -> 1
reduce принимает на вход функцию, принимающую либо 0, либо 2 аргумента.
Если reduce передана бинарная функция fn, то
(reduce #'fn '(arg1 arg2 arg3 ... argN)) <=>
<=> (fn (... (fn (fn arg1 arg2) arg3) ...) argN)
Пример:
(reduce #'list '(1 2 3 4)) ; (((1 2) 3) 4)

Если reduce передана функция fn, не принимающая аргументов, то
(reduce #'fn '()) <=> (fn)
Пример:
(reduce #'list '()) ; (list) = NIL
(reduce #'+ '()) ; (+) = 0
(reduce #'* '()) ; (*) = 1

Такое поведение может быть полезно, когда нужно гарантировать, что в случае передачи пустой последовательности, будет возвращено значение по умолчанию, определённое логикой функции.

Если reduce передан список из одного элемента, то reduce вернёт этот элемент вне зависимости от переданной функции, например:
(reduce #'reduce '(1)) ; 1
```

```
; 9. * Пусть list-of-list список, состоящий из списков. Написать функцию, которая вычисляет сумму длин всех элементов
   list-of-list (количество атомов), т.е. например для аргумента
   ((1 2) (3 4)) -> 4.
(defun atom-count-helper (lst cnt)
  (cond ((null lst) cnt)
        ((atom (car lst)) (atom-count-helper (cdr lst) (+ 1 cnt)))
        (t (atom-count-helper (cdr lst) cnt))))
(defun ac (lst)
  (atom-count-helper lst 0))

(defun f9-helper (lol cnt)
  (cond ((null lol) cnt)
        (t (f9-helper (cdr lol) (+ cnt (ac (car lol)))))))

(defun f9 (lol)
  (f9-helper lol 0))
```