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, искусственный интелект и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчёт по лабораторной работе № 4 по курсу «Функциональное и логическое программирование»

Тема Использование управляющих структур, работа со списками

Студент Волков Г. В.

Группа ИУ7-61Б

Оценка (баллы)

Преподаватель Толпинская Н. Б.

Москва — 2023 г.

Практические задания

Задание 1

Чем принципиально отличаются функции cons, list, append?

cons является базовой функцией. list и append реализованы через cons.

cons является чистой функцией и принимает 2 параметра. Она создаёт списочную ячейку, в которой car указывает на первый элемент, а cdr на второй.

list является формой, так как принимает произвольное количество аргументов. Возвращает список из аргументов.

append является формой, так как принимает произвольное количество аргументов. Возвращает конкатенацию аргументов. Она возвращает точечную пару, car указывает на конкатенацию всех переданных аргументов, кроме последнего, а cdr на последний аргумент.

Пусть (setf lst1 '(a b c))
(setf lst2 '(d e)).

Каковы результаты вычисления следующих выражений?

Листинг 1 – Задание 1

```
(setf lst1 '( a b c))
(setf lst2 '( d e))

(cons lstl lst2) ; -> ((a b c) d e)
(list lst1 lst2) ; -> ((a b c) (d e))
(append lst1 lst2) ; -> (a b c d e)
```

Задание 2

Каковы результаты вычисления следующих выражений, и почему?

Листинг 2 – Задание 2

```
(reverse '(a b c)) ; -> (c b a)
(reverse ()) ; -> Nil
(reverse '(a b (c (d)))) ; -> ((c (d)) b a)
(reverse '((a b c))) ; -> (( a b c))
(reverse '(a)) ; -> (a)
(last '(a b c)) ; -> (c)
(last '(a b (c))) ; -> ((c))
(last '(a)) ; -> (a)
(last ()) ; -> Nil
(last '((a b c))) ; -> ((a b c))
```

Задание 3

Написать, по крайней мере, два варианта функции, которая возвращает последний элемент своего списка-аргумента.

Листинг 3 – Задание 3

```
(defun last1 (lst) (car (last lst)))
(defun last2 (lst) (car (reverse lst)))
(defun last3 (lst)
    (if (cdr lst)
        (last3 (cdr lst))
        (car lst)
    )
)
```

Задание 4

Написать, по крайней мере, два варианта функции, которая возвращает свой список аргумент без последнего элемента.

Листинг 4 – Задание 4

```
(defun remove_last1 (lst) (reverse (cdr (reverse lst))))
(defun remove_last2 (lst) (reverse (last (reverse lst) (- (length lst) 1))))
(defun remove_last3 (lst)
    (if (cdr lst)
        (cons
            (car lst)
            (remove_last3 (cdr lst))
        )
    )
)
```

Задание 5

Напишите функцию swap-first-last, которая переставляет в списке аргументе первый и последний элементы.

Листинг 5 – Задание 5

```
(defun swap-first-last (lst)
    (cons
        ((null lst) Nil)
        ((= (length lst) 1) lst)
        (T
            (let ((f (car lst)))
                (setf (car lst) (car (last lst)))
                (setf (car (last lst)) f)
                lst
            )
        )
    )
)
```

Задание 6

Написать простой вариант игры в кости, в котором бросаются две правильные кости. Если сумма выпавших очков равна 7 или 11 — выигрыш, если выпало (1,1) или (6,6) — игрок имеет право снова бросить кости, во всех остальных случаях ход переходит ко второму игроку, но запоминается сумма выпавших очков. Если второй игрок не выигрывает абсолютно, то выигрывает тот игрок, у которого больше очков. Результат игры и значения выпавших костей выводить на экран с помощью функции print.

Листинг 6 – Задание 6

```
(defun roll ()
    (cons
        (+ (random 6) 1)
        (+ (random 6) 1)
    )
)

(defun check_reroll (pair)
    (or
        (= (car pair) (cdr pair) 1)
        (= (car pair) (cdr pair) 6)
    )
)

(defun roll_sum (pair)
    (+ (car pair) (cdr pair))
)

(defun check_win (pair)
    (or
        (= (roll_sum pair) 7)
        (= (roll_sum pair) 11)
    )
)

(defun play_round ()
    (let ( (pair (roll)) )
        (cond
            (
                (check_reroll pair)
                (play_round)
            )
            (T pair)
```

```
        )
    )
)

(defun play_dice ()
    (let ( (pl1 (play_round)) (pl2 (play_round)) )
        (format T "~%Player 1: ~A ~%" pl1)
        (format T "Player 2: ~A ~%" pl2)
        (cond
            (
                (check_win pl1)
                (princ "Player 1 wins")
            )
            (
                (check_win pl2)
                (princ "Player 2 wins")
            )
            (
                (> (roll_sum pl1) (roll_sum pl2))
                (princ "Player 1 wins")
            )
            (
                (< (roll_sum pl1) (roll_sum pl2))
                (princ "Player 2 wins")
            )
            (
                T
                (princ "Draw")
            )
        )
    )
)
```

Задание 7

Написать функцию, которая по своему списку-аргументу lst определяет является ли он палиндромом (то есть равны ли lst и (reverse lst)).

Листинг 7 – Задание 7

```
(defun is_palindrome (lst)
    (cond
        ((< (length lst) 2) T)
        ((equalp (car lst) (car (last lst)))
        (is_palindrome (cdr (reverse (cdr lst)))))
        (T Nil)
    )
)
```

Задание 8

Напишите свои необходимые функции, которые обрабатывают таблицу из 4-х точечных пар: (страна . столица), и возвращают по стране - столицу, а по столице — страну.

Листинг 8 – Задание 8

```
(defun find_in_table (lst item)
    (cond
        ((null lst) Nil)
        ((equal item (caar lst)) (cdar lst))
        ((equal item (cdar lst)) (caar lst))
        (T (find_in_table (cdr lst) item))
    )
)
```

Задание 9

Напишите функцию, которая умножает на заданное число-аргумент первый числовой элемент списка из заданного 3-х элементного списка аргумента, когда

а) все элементы списка — числа,

б) элементы списка – любые объекты.

Листинг 9 – Задание 9

```
(defun mult_first_num (num lst)
    (cond
        ((null lst) Nil)
        ((numberp (car lst)) (cons (setf (car lst) (* (car lst) num)) (cdr lst)))
        (T (cons (car lst) (mult_first_num num (cdr lst))))
    )
)
```