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 № 4 по курсу «Функциональное и логическое программирование»

Тема Использование управляющих структур, работа со списками

Студент Виноградов А. О.

Группа ИУ7-66Б

Оценка (баллы)

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

Москва — 2023 г.

Введение

Цель работы: приобрести навыки работы с управляющими структурами Lisp.

Задачи работы: изучить работу функций с произвольным количеством аргументов, функций разрушающих и неразрушающих структуру исходных аргументов.

1 Практические задания

1.1 Чем принципиально отличаются функции cons, list, append? Пусть (setf lst1 '(a b c)) (setf lst2 '(d e)). Каковы результаты вычисления следующих выражений?

```
1 (cons lst1 lst2) ;((a b c) d e)
2 (list lst1 lst2) ;((a b c) (d e))
3 (append lst1 lst2) ; (a b c d e)
```

1.2 Каковы результаты вычисления следующих выражений, и почему?

```
1  (reverse '(a b c)) ;(c b a)
2  (reverse ()) ;Nil
3  (reverse '(a b (c (d)))) ;((c (d)) b a)
4  (reverse '((a b c))) ;((a b c))
5  (reverse '(a)) ;(a)
6  (last '(a b c)) ;(c)
7  (last '(a b (c))) ;((c))
8  (last '(a)) ;(a)
9  (last ()) ;Nil
10 (last '((a b c))) ;((a b c))
```

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

```
(defun f3 (x)
    (car (last x))
)
```

```
(defun f32 (lst)
    (cond
        ((cdr  lst) (f32  (cdr lst)))
        (T(car lst))
    )
)
```

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

```
(defun without_last (x)
    (
        cond
        ((cdr x) (cons (car x) (without_last (cdr x))))
        (T Nil)
    )
)
```

```
(defun without_last (x)
    (reverse (cdr (reverse x)))
)
```

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

```
(defun swap-first-last (x)
    (nconc
        (last x)
        (without_last (cdr x))
        (cons (car x) Nil)
    )
)
```

1.6

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 1.1 – sx.lsp

```
(defun sum (throw)
    (+ (car throw) (cdr throw)))

(defun 1_roll()
    (+ (random 6) 1))

(defun roll()
    (cons (1_roll) (1_roll)))

(defun reroll_d (sum_count)
    (or (= sum_count 2) (= sum_count 12)))

(defun win_d (sum_count)
    (or (= sum_count 7) (= sum_count 11)))

```

```
16
17
18 (defun turn (&aux (dices (roll)))
19     (terpri)
20     (princ "Rolled:")
21     (princ dices)
22
23     (cond
24         ((win_d (sum dices)) Nil)
25         ((not (reroll_d (sum dices))) (sum dices))
26         (T (turn))
27     )
28 )
29
30 (defun second_turn (roll1 &aux (roll2 (turn)))
31     (terpri)
32     (princ "Second␣player's␣turn␣ended")
33     (terpri)
34     (terpri)
35
36     (if roll2
37         (cond
38             ((> roll1 roll2) "First␣player␣wins!")
39             ((= roll1 roll2) "Draw!")
40             (T "Second␣player␣wins!")
41         )
42         "Second␣player␣wins!"
43     )
44 )
45
46 (defun game (&aux (roll1 (turn)))
47     (terpri)
48     (princ "First␣player's␣turn␣ended")
49     (terpri)
50     (if roll1
51         (second_turn roll1)
52         "First␣player␣wins!"
53     )
54 )
```

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

```
(defun is_palindrom (lst)
    (equalp lst (reverse lst))
)
```

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

```
(defun show_on_map (lst item)
    (cond
        ((equal (caar lst) item) (cdar lst))
        ((equal (cdar lst) item) (caar lst))
        ((cdr lst) (show_on_map (cdr lst) item))
    )
)
```

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

Когда: a) все элементы списка — числа, b) элементы списка – любые объекты.

```
(defun mul_a (lst num)
    (cond
        (
            (and
                (numberp num)
                (numberp (car lst))
                (numberp (cadr lst))
                (numberp (caddr lst)))
        ) (* (car lst) num)
        (T Nil)
    )
)
```

```
(defun mul_b (lst num)
    (cond
        ((and (numberp (car lst)) (numberp num)) (* (car lst) num))
        ((cdr lst) (mul_first_num (cdr lst) num))
        (T Nil)
    )
)
```