Государственное образовательное учреждение высшего профессионального образования

«Московский государственный технический университет имени Н. Э. Баумана»

(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Отчет

По лабораторной работе №7

По курсу «Функциональное и логическое программирование»

Студент: Киселев А.М.

Группа: ИУ7-66

Преподаватель: Толпинская Н.Б.

Москва 2019

Содержание

1 Выполнение работы

1.1 Первое задание

Написать предикат, который принимает два числа-аргумента и возвращает T, если первое число не меньше второго.

Данная функция представлена в 1.1

Листинг 1.1 — Предикат, который проводит сравнение чисел a и b.

```
(defun f (a b) (>= a b))
```

1.2 Второе задание

Переписать функцию how-alike, приведенную в лекции и использующую COND, используя конструкции IF, AND/OR.

1.3 Третье задание

Чем принципиально отличаются функции cons, list, append?

Пусть (setf lst1 ‘(a b))(setf lst2 ‘(c d))

Каковы результаты следующих выражений? Сами выражения и их результаты представленны в 1.2

Принципиальные отличия cons и list заключаются в:

а) количестве аргументов(cons принимает только два аргумента, а list - неограниченное количество;

б) в конечном результате. Cons возвращает точечную пару, которая в итоге может оказаться списком. List - только список.

Принципиальное отличие append от list:

а) в аргументах, подающихся на вход. Append принимает на вход только списки. List может принимать как списки, так и атомы;

Листинг 1.2 — Выражения и их результата.

```
(cons lst1 lst2)
;((a b) c d)
(list lst1 lst2)
;((a b) (c d))
(append lst1 lst2)
;(a b c d)
```

1.4 Четвертое задание

Каковы результаты вычисления следующих выражений?
Сами выражения и их результаты представлены в 1.3

Листинг 1.3 — Выражения и их результата.

```
(reverse ())
;Nil
(last ())
;Nil
(reverse `(a))
;(a)
(last `(a))
;(a)
(reverse `((a b c)))
;((a b c))
(last `((a b c)))
;((a b c))
```

1.5 Пятое задание

Написать, по крайней мере, два варианта функции, которая возвращает последний элемент своего списка-аргумента.
Функции представлены в 1.4

Листинг 1.4 — Функции, возвращающие последний элемент списка

```
(defun f1(lst)
    (if (null lst)
        lst
        (or (f1 (cdr lst))
            (if (null (cdr lst))
                lst
            )
        )
    )
)

(defun f2(lst)
    (car (reverse lst))
)
```

1.6 Шестое задание

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

Функции представлены в 1.5

Листинг 1.5 — Функции, возвращающие список без последнего элемента

```
(defun f1(lst)
        (if (null lst)
            lst
            (if (not (null (cdr lst)))
                (cons (car lst) (f1 (cdr lst)))
            )
        )
)

(defun f2(lst)
    (reverse (cdr (reverse lst)))
)
```

1.7 Седьмое задание

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

Листинг программы представлен в 1.6

Листинг 1.6 — Программа симмуляции игры в кости.

```

(setf *random-state* (make-random-state t))

(defun throw-dice()
    (list (+ 1 (random 6)) (+ 1 (random 6)))
)

(defun player-throw(name)
```

```
    (setf thrw (throw-dice))
    (print name)
    (princ ": ")
    (write thrw)
    (if (or (equal thrw '(1 1))
        (equal thrw '(6 6))
        )
        (player-throw name)
    )
    thrw
)

(defun dice-sum(thrw)
    (+ (car thrw) (cadr thrw))
)

(defun abs-win(sum)
    (if (or (eq sum 7) (eq sum 11))
        T
        Nil
    )
)

(defun game()
    (setf sum1 (dice-sum (player-throw 'Player1)))
    (setf res '(Player1 won!))

    (if (not (abs-win sum1))
        (progn
            (setf sum2 (dice-sum (player-throw 'Player2)))
            (if (eq sum1 sum2)
                (game)
                (if (> sum2 sum1)
                    (setf res '(Player2 won!))
                )
            )
        )
        ()
    )
    res
)
```