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

к лабораторной работе №6

По курсу: «Функциональное и логическое программирование»

Тема: «Использование управляющих структур, работа со списками».

Студент: Якуба Д.В.
Группа: ИУ7-63Б
Преподаватели: Толпинская Н. Б.,
Строганов Ю. В.

Москва, 2021 г.

Практическая часть

Задание 1. Чем принципиально отличаются функции cons, list, append?
Пусть (setf lst1 `(a b)) (setf lst2 `(c d))
Каковы результаты вычисления следующих выражений?
1. (cons lst1 lst2);
2. (list lst1 lst2);
3. (append lst1 lst2);

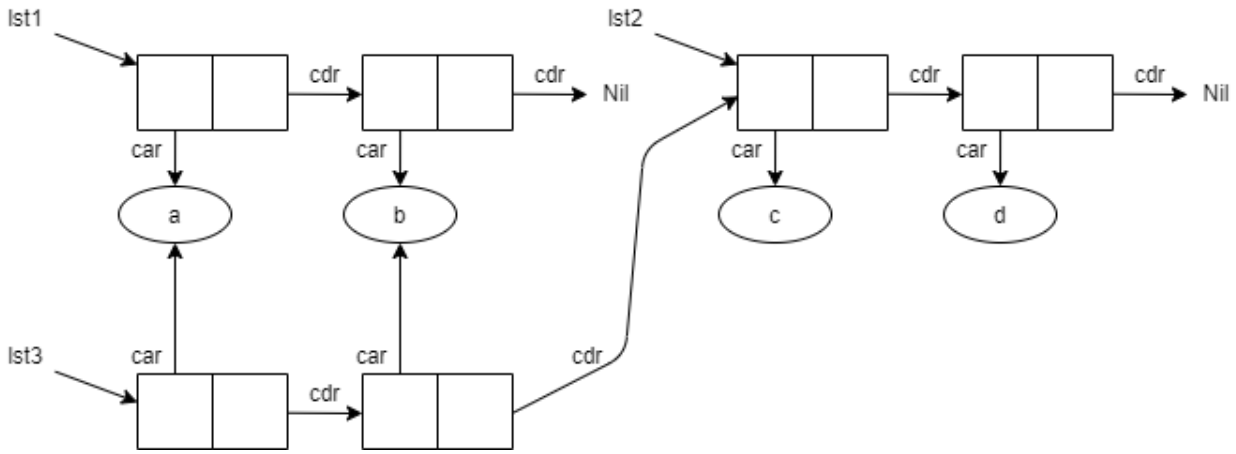
Ответ:

Cons объединяет значения двух своих аргументов в точечную пару. В случае, если второй переданный аргумент является списком, то результатом выполнения будет список.

List создаёт список, включающий в себя все переданные функции аргументы.

Функция append объединяет элементы списков-аргументов, которые расположены на самом верхнем уровне. При этом стоит отметить факт того, что «копируется» только первый переданный список, который будет идти перед вторым указанным списком. Таким образом, если второй переданный список будет модифицирован, итоговый список также может быть изменён (см. изображение ниже).

(setf lst3 (append lst1 lst2))



1. (cons lst1 lst2) -> ((A B) C D);

2. (list lst1 lst2) -> ((A B) (C D));

3. (append lst1 lst2) -> (A B C D);

Задание 2. Каковы результаты вычисления следующих выражений?

1. (reverse());

2. (last());

3. (reverse `(a));

4. (last `(a));

5. (reverse `((a b c)));

6. (last `((a b c)));

Ответ:

1. (reverse()) -> NIL;

2. (last()) -> NIL;

3. (reverse `(a)) -> (A);

4. (last `(a)) -> (A);

5. reverse `((a b c))) -> ((A B C)). «Порядок» списка не был изменён на втором уровне (на уровне элемента-списка (A B C), так как функция reverse, как и большинство функций в LiSP, работают только с первым уровнем списка. Таким образом, обратным порядком одноэлементного списка является сам одноэлементный список;

6. (last `((a b c))) -> (A B C). Функция last также работает только с первым уровнем переданного ей списка;

Задание 3. Написать, по крайней мере, два варианта функции, которая возвращает последний элемент своего списка-аргумента.

Решение:

```
; с использованием функции reverse
(defun retLast (lst) (
    car (reverse lst)
))

; с использованием рекурсии
(defun retLast (lst) (
    if (null (cdr lst))
    (car lst)
    (retlast (cdr lst))
))

; с возвращением последнего элемента любого уровня
(defun retLastEnd (lst) (
    if (null (cdr lst))
    (let ((tempLst (car lst)))
        (if (listp tempLst)
        (retLastEnd tempLst)
        tempLst
        )
    )
    (retLastEnd (cdr lst))
))
```

Задание 4. Написать, по крайней мере, два варианта функции, которая возвращает свой список-аргумент без последнего элемента.

Решение:

```
; с использованием функции reverse
(defun retWOLast (lst) (
    reverse (cdr (reverse lst))
))

; с использованием рекурсии
(defun retWOLast (lst) (
    if (null (cdr lst))
    nil
    (cons (car lst) (retWOLast (cdr lst)))
))
```

Задание 5. 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 получает 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.

Решение:

```
(defun rollTheDices (numberOfEdges)
    (let ((sum (+ (random numberOfEdges) (random numberOfEdges) 2))) ; + 2 так ка
к random генерируется число от нуля
    (and
        (print (list 'Сумма 'из 'броска sum))
        (if (or (= sum 2) (= sum 12))
            (and
            (print '(Игрок получил возможность перебросить кости))
            (setq sum (rollTheDices numberOfEdges)))
            sum)
        sum)))

(defun isWinByNumber (roll)
    (or (= roll 7) (= roll 11)))

(defun whoWon (fPlayerRoll sPlayerRoll)
    (or
        (if (isWinByNumber fPlayerRoll) '(Игрок 1 выиграл))
        (if (or (isWinByNumber sPlayerRoll) (> sPlayerRoll fPlayerRoll)) '(Игрок
2 выиграл))
        (if (> fPlayerRoll sPlayerRoll) '(Игрок 1 выиграл))
        'Ничья))

(defun playTheGame ()
    (print (whoWon (and
                        (print '(Игрок 1 бросает кости))
```

```
        (rollTheDices 6))
(and
        (print '(Игрок 2 бросает кости))
        (rollTheDices 6)))))
```

Теоретическая часть

1. Структуроразрушающие и не разрушающие структуру списка функции.

Структуроразрушающими называются функции, при помощи которых можно вносить изменения во внутреннюю структуру уже существующих выражений.

Пример структуроразрушающей функции – rplaca.

Пример не разрушающей структуры списка функции – append.

2. Отличия в работе функций cons, list, append и в их результате.

Cons объединяет значения двух своих аргументов в точечную пару. В случае, если второй переданный аргумент является списком, то результатом выполнения будет список.

List создаёт список, включающий в себя все переданные функции аргументы. List организован с использованием cons.

Функция append объединяет элементы списков-аргументов, которые расположены на самом верхнем уровне. При этом стоит отметить факт того, что «копируется» только первый переданный список, который будет идти перед вторым указанным списком. Таким образом, если второй переданный список будет модифицирован, итоговый список также может быть изменён (см. изображение ниже).

(setf lst3 (append lst1 lst2))