|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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»\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Лабораторная работа №7*

*По предмету: «Функциональное и логическое программирование»*

Преподаватель: Толпинская Н.Б.,

Студент: Мирзоян С.А.,

Группа: ИУ7-65Б

Москва, 2020 г.

**1. Чем принципиально отличаются функции cons, list, append?**

*cons* - создаёт одну списочную ячейку. CAR-указатель ячейки указывает на первый аргумент, CDR-указатель - на второй.

*list -* создаёт список, в котором car-указатель i-й списочной ячейки указывает на i-й аргумент.

*append* - копирует все аргументы, кроме последнего, и создает из них список по следующему принципу: CDR-указатель последнего элемента i-й копии указывает на первый элемент (i+1)-й копии. CDR-указатель последней копии указывает на последний (не скопированный) аргумент.

**Пусть**

**(setf lst1 '( a b))**

**(setf lst2 '( c d))**

**Каковы результаты вычисления следующих выражений:**

*(cons lst1 lst2) ->* ((A B) C D)

*(list lst1 lst2) ->* ((A B) (C D))

*(append lst1 lst2) ->* (A B C D)

**2. Каковы результаты вычисления следующих выражений:**

*(reverse ‘()) -> NIL  
(last ()) -> NIL  
(reverse '(a)) -> A  
(last '(a)) -> A  
(reverse '((a b c))) -> (A B C)  
(last '((a b c))) -> (A B C)*

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

1. (defun flast\_1 (x) (car (reverse x)))
2. (defun f\_last2 (x)  
    (if (equal (cdr x) nil)

(car x)

(mylast2 (cdr x))

)

)

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

1. (defun fun (x) (reverse (cdr (reverse x))))
2. (defun fun (x)

(cons (car x)

(if (> (length (cdr x)) 1)

(cut2 (cdr x))

nil

)

)

)

**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 абслютно, то выигрывает тот игрок, у которого больше очков. Результат игры и значения выпавших костей выводить на экран с помощью функции print.**

;Бросок двух игральных костей

(defun first\_roll () (print (list (+ (random 6) 1) (+ (random 6) 1))))

;Проверка, будет ли дополнительный бросок. Если да,

(defun second\_roll (x)

(if (or (equal x '(1 1))

(equal x '(6 6)))

(and (print "Повторный бросок") (first\_roll))

x

)

)

;Ход(ы) одного игрока

(defun roll () (second\_roll (first\_roll)))

;сумма

(defun summary (r) (+ (first r) (second r)))

;Проверка на победителя

(defun winner (res)

(if (or (= res 7)

(= res 11)

)

t

nil

)

)

(defun play ()

(cond ( (and (print "Первый игрок:")

(winner (setq p1 (summary (roll))))

)

"Первый игрок победил "

)

( (and (print "Second player:")

(winner (setq p2 (summary(roll))))

)

"Второй игрок победил"

)

((= p1 p2) "Ничья!") ; Равное количество очков

((> p1 p2) "Первый игрок победил"); Если при дополнительных бросках сумма баллов первого игрока больше.

(t "Второй игрок победил"); иначе

)

)