|  |  |
| --- | --- |
| 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 **ПРОГРАМНОЕ ОБЕСПЕЧЕНИЕ ЭВМ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.03 ПРОГРАММНАЯ ИНЖЕНЕРИЯ**

**Отчет**

|  |  |
| --- | --- |
| **По лабораторной работе №** | 6 |

**Дисциплина:** Функциональное и логическое программирование

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ7-62Б |  |  | Н.А. Гарасев |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Н.Б.Толпинская |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2021

1. **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)

1. **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))

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

(defun my\_last\_1 (lst) (car (reverse lst)))

(my\_last\_1 `( 1 2 3 4 5 6 7 8)) – 8

(my\_last\_1 `(1)) – 1

(my\_last\_1 `()) – NIL

(defun my\_last\_2 (lst) (nth (- (length lst) 1) lst))

(my\_last\_2 `(1 2 3 4 5 6 7 8)) – 8

(my\_last\_2 `(1)) – 1

(my\_last\_2 `()) – NIL

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

(defun non\_cdr\_1 (lst) (reverse (cdr (reverse lst))))

(non\_cdr\_1 `(1 2 3 4 5 6)) - (1 2 3 4 5)

(non\_cdr\_1 `(1)) – NIL

(non\_cdr\_1 `()) – NIL

(defun non\_cdr\_2 (lst) (reverse (nthcdr 1 (reverse lst))))

(non\_cdr\_2 `(1 2 3 4 5 6 7 8 9)) - (1 2 3 4 5 6 7 8)

(non\_cdr\_2 `(1)) – NIL

(non\_cdr\_2 `()) – NIL

1. **Написать простой вариант игры в кости.**

**(defun** roll **()** **(+** 1 **(random** 6**)))**

**(defun** roll\_dice **(**n**)**

**(let** **((**x **(**roll**))**

**(**y **(**roll**)))**

**(and** **(**print **`(**roll ,n**))** **(**print x**)** **(**print y**)**

**(cond** **((and** **(eq** x y**)** **(or** **(eq** x 1**)** **(eq** x 6**)))** **(**roll\_dice n**))**

**((or** **(eq** **(+** x y**)** 7**)** **(eq** **(+** x y**)** 11**))** T**)**

**(**T **(+** x y**))))))**

**(defun** play\_1 **()**

**(let** **((**x **(**roll\_dice 1**))**

**(**y **(**roll\_dice 2**)))**

**(cond** **((eq** x T**)** **`(**First player WIN**))**

**((eq** y T**)** **`(**Second player WIN**))**

**((>** x y**)** **`(**First player WIN**))**

**((eq** x y**)** **`(**TIE**))**

**(**T **`(**Second player WIN**)))))**

**(defun** play\_2 **()**

**(let** **((**x **(**roll\_dice 1**)))**

**(cond** **((eq** x T**)** **`(**First player WIN**))**

**((let** **((**y **(**roll\_dice 2**)))**

**(cond** **((eq** y T**)** **`(**Second player WIN**))**

**((>** x y**)** **`(**First player WIN**))**

**((eq** x y**)** **`(**TIE**))**

**(**T **`(**Second player WIN**))))))))**

**Вопросы.**

**Вопрос 1. Структуроразрушающие и не разрушающие структуру списка функции.**

Функции делятся на два типа: структуроразрушающие и не разрушающие структуру. Первый тип функций используют структуры, которые им переданы, а второй тип использует копии, там, где нельзя иначе. Например append – не разрушает структуру, делает копию всех аргументов, кроме последнего. Другая функция nconc – также объединяет структуры, но заставляет последний элемент первого аргумента ссылаться на второй аргумент и т.д.

**Вопрос 2. Отличие в работе функций cons, list и append и в их реализации.**

Функция cons принимает только 2 аргумента. Функция создает бинарный узел, указатели которого достаются двум входным аргументам и возвращает точечную пару. Функция list принимает переменное число аргументов. Функция создает столько бинарных узлов, сколько было передано ей аргументов и возвращает список из этих аргументов. Функция append создает копии всех аргументов, кроме первого, затем второй указатель последнего элемент копии первого аргумента ссылается на первый элемент копии второго аргумента и т.д.