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 №6 по курсу «Функциональное и логическое программирование»

Тема Рекурсивные функции

Студент Кононенко С.С.

Группа ИУ7-63Б

Оценка (баллы) _____

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

Москва — 2021 г.

Задание 1

Постановка задачи. Каковы результаты вычисления следующих выражений?

Решение.

Листинг 1 – Решение задания 1

```
(setf lst1 '(a b))
(setf lst2 '(c d))
(cons lst1 lst2) ; ((A B) C D)
(list lst1 lst2) ; ((A B) (C D))
(append lst1 lst2) ; (A B C D)
```

Задание 2

Постановка задачи. Каковы результаты вычисления следующих выражений?

Решение.

Листинг 2 – Решение задания 2

```
(reverse ()) ; NIL
(last ()) ; NIL
(reverse '(a)) ; (A)
(last '(a)) ; (A)
(reverse '((a b c))) ; ((A B C))
(last '((a b c))) ; ((A B C))
```

Задание 3

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

Решение.

Листинг 3 – Решение задания 3

```
(defun make-last-rev (lst)
```

```
    (car (reverse lst)))
(defun make-last-rec (lst)
    (if (null (cdr lst)) (car lst) (make-last-rec (cdr lst))))
```

Задание 4

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

Решение.

Листинг 4 – Решение задания 4

```
(defun make-notail-rev (lst)
    (nreverse (cdr (reverse lst))))
(defun make-notail-rec (lst)
    (if (null (cdr lst)) Nil (cons (car lst) (make-notail-rec (cdr lst)))))
```

Задание 5

Постановка задачи. 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.

Листинг 5 – Решение задания 5

```
(defun roll-dices (edges)
    (let ((sum (+ (random edges) (random edges) 2)))
    (and
        (print (list 'Points '= sum))
        (if (or (= sum 2) (= sum 12))
            (and (print '(Reroll chance)) (setq sum (roll-dices edges)))
            sum)
        sum)))
```

```
(defun is-early-win (points)
    (or (= points 7) (= points 11)))
(defun make-result (fpoints spoints)
    (cond ((or (is-early-win fpoints) (> fpoints spoints)) '(First player won))
        ((or (is-early-win spoints) (> spoints fpoints)) '(Second player won))
        (T 'Draw)))
(defun make-game ()
    (print (make-result (and (print '(Player 1 rolls)) (roll-dices 6))
                        (and (print '(Player 2 rolls)) (roll-dices 6)))))
```

Ответы на контрольные вопросы

Вопрос 1. Структуроразрушающие и не разрушающие структуру списка функции.

Ответ.

Не разрушающие структуру функции

Данные функции не меняют сам объект-аргумент, а создают копию.

Функция `append`

Объединяет списки. Это форма, можно передать больше 2 аргументов. Создает копию для всех аргументов, кроме последнего.

Пример: `(append '(1 2) '(3 4))` — `(1 2 3 4)`.

Функция `reverse`

Возвращает копию исходного списка, элементы которого переставлены в обратном порядке. **В целях эффективности работает только на верхнем уровне**.

Пример: `(reverse '(1 2 3 4))` — `(4 3 2 1)`.

Функция `last`

Проход по верхнему уровню и возврат последней списковой ячейки.

Пример: `(last '(1 2 3 4))` — `(4)`.

Функция `nth`

Возврат указателя от n-ной списковой ячейки, нумерация с нуля.

Пример: `(nth 1 '(1 2 3 4))` — 2.

Функция `nthcdr`

Возврат n-ого хвоста.

Пример: `(nthcdr 1 '(1 2 3 4))` — `(2 3 4)`.

Функция `length`

Возврат длины списка (**только по верхнему уровню**).

Пример: `(length '(1 2 (3 4)))` — `3`.

Функция `remove`

Модифицирует, но работает с копией, поэтому не разрушает. Данная функция удаляет элемент по значению (Часто разрушающая аналогичная функция называется `delete`). По умолчанию используется `eql` для сравнения на равенство, но можно передать другую функцию через ключевой параметр `:test`.

Пример: `(remove 3 '(1 2 3))` — `(1 2)`;

Функция `rplaca`

Переставляет `car`-указатель на 2 элемент-аргумент (S-выражение).

Пример: `(rplaca '(1 2 3) 3)` — `(3 2 3)`.

Функция `rplacd`

Переставляет `cdr`-указатель на 2 элемент-аргумент (S-выражение).

Пример: `(rplacd '(1 2 3) '(4 5))` — `(1 4 5)`.

Функция `subst`

Заменяет все элементы списка, которые равны 2 переданному элементу-аргументу на другой 1 элемент-аргумент. *По умолчанию для сравнения используется функция* `eql`.

Пример: `(subst 2 1 '(1 2 1 3))` — `(2 2 2 3)`.

Структуроразрушающие функции

Данные функции меняют сам объект-аргумент, невозможно вернуться к исходному списку. Чаще всего такие функции начинаются с префикса `n-`.

Функция `nconc`

Работает аналогично `append`, только не копирует свои аргументы, а разрушает структуру.

Функция `nreverse`

Работает аналогично `reverse`, но не создает копии.

Функция `nsubst`

Работае аналогично функции `nsubst`, но не создает копии.

Вопрос 2. Отличие в работе функций `cons`, `list`, `append` и в их результате.

Ответ.

Функция `cons` — чисто математическая, конструирует списковую ячейку, которая может вовсе и не быть списком (будет списком только в том случае, если 2 аргументом передан список).

Примеры:

1. `(cons 2 '(1 2))` — `(2 1 2)` — список;

2. `(cons 2 3)` — `(2 . 3)` — не список.

Функция `list` — форма, принимает произвольное количество аргументов и конструирует из них список. Результат — всегда список. При нуле аргументов возвращает пустой список.

Примеры:

1. `(list 1 2 3)` — `(1 2 3)`;

2. `(list 2 '(1 2))` — `(2 (1 2))`;

3. `(list '(1 2) '(3 4))` — `((1 2) (3 4))`;

Функция `append` — форма, принимает на вход произвольное количество аргументов и для всех аргументов, кроме последнего, создает копию, ссылая при этом последний элемент каждого списка-аргумента на первый элемент следующего по порядку списка-аргумента (так как модифицируются все списки-аргументы, кроме последнего, копирование для последнего не делается в целях эффективности).

Примеры:

1. `(append '(1 2) '(3 4))` — `(1 2 3 4)`;

2. `(append '((1 2) (3 4)) '(5 6))` — `((1 2) (3 4) 5 6)`.