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»

Отчёт

по лабораторной работе №4

Название: Использование управляющих структур, работа со списками

Дисциплина: Функциональное и логическое программирование

Студент	ИУ7-64Б		Л.Е.Тартыков
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			Н.Б.Толпинская
		(Подпись, дата)	(И.О. Фамилия)
Преподаватель			Ю.В.Строганов
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2022

1 Теоретические вопросы

1.1 Синтаксическая форма и хранение программы в памяти

В Lisp программы и данные синтаксически выглядят в виде *S-выражения*. S-выражение может быть как атомом, который в памяти представляется в виде 5 указателей (name, value, function, properties, package), так и точечной парой – 2 указателя (бинарный узел).

1.2 Трактовка элементов списка

Элементы списка представляются следующим образом: первый - имя функции, остальные – её аргументы. Формат списка представлен в виде формулы (1.1).

$$(\text{функция аргумент\_1} \ldots \text{аргумент\_n}), n \geq 0 \tag{1.1}$$

1.3 Порядок реализации программы

Набранные S-выражения выполняются при помощи интерпретатора – функцией eval, после выполнения которой возвращается полученный результат.

1.4 Способы определения функции

1. lambda-выражение. Данный способ представлен в виде формулы (1.2).

$$(\textit{lambda}\ \lambda\text{-список форма}), \tag{1.2}$$

где λ-список – список формальных параметров, форма – тело функции.

lambda-выражение не хранится в памяти и не имеет имени. Вычисляется сразу же. Используется для повторных вычислений.

Вызов lambda-функции выполняется по формуле (1.3).

$$(\lambda\text{-выражение последовательность форм}) \tag{1.3}$$

2. С помощью *defun* по формуле (1.4).

$$(defun\ f\ \lambda\text{-выражение}) \tag{1.4}$$

Система по имени символьного атома находит его определение.

1.5 Принципиальное отличие функций cons, list, append

2 Практические задания

2.1

```
(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)
```

2.2 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))
```

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

```
(defun last_elem_1 (arg) (car (last arg)))

(defun last_elem_2 (arg) (first (reverse arg)))
```

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

```
1 (defun without_last_1 (arg) (reverse (cdr (reverse arg))))
```

2.5 Игра в кости

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 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.