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 (ИУ7)**

ОТЧЕТ

по лабораторной работе № 5

Название: Списки в LISP.

Дисциплина: Функциональное и логическое программирование

Студент ИУ7-63Б _____ В.П. Федоров
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ Н.Б. Толпинская
(Подпись, дата) (И.О. Фамилия)

Москва, 2021

Цель работы:
изучить правила и особенности работы функций: cond, if, and, or, append, remove, reverse, substitute и др.

Задача 1.

Написать функция, которая принимает целое число и возвращает первое четное число, не меньшее аргумента.

Листинг 1: решение задачи №1.

```
(defun get_first_greater_or_eq_even_num(num)
	(cond
		((= (mod num 2) 0) num)
		(T (+ num 1))))
```

Задача 2.

Написать функция, которая принимает число и возвращает число того же знака, но с модулем на 1 больше модуля аргумента.

Листинг 2: решение задачи №2.

```
(defun get_lager_in_modules_num(num)
	(cond
		((>= num 0) (+ num 1))
		((< num 0) (- num 1))))
```

Задача 3.

Написать функция, которая принимает два числа и возвращает список из этих чисел, расположенный по возрастанию.

Листинг 3: решение задачи №3.

```
(defun get-rising-list(numOne numTwo)
	(cond
		((>= numOne numTwo) (list numTwo numOne))
		((<  numOne numTwo) (list numOne numTwo))))
```

Задача 4.

Написать функция, которая принимает три числа и возвращает T только тогда, когда первое число расположено между вторым и третьим

Листинг 4: решение задачи №4.

```
(defun check-sequence(numOne numTwo numThree)
        (cond
                ((AND (>= numOne numTwo) (<= numOne numThree)) T))
```

.

Задача 5.

Каков результат вычисления следующих выражений?

выражение	**результат**
(and ‘fee ‘fie ‘foe)	FOE
(or ‘fee ‘fie ‘foe)	FEE
(and (equal ‘abc ‘abc) ‘yes)	YES
(or nil ‘fie ‘foe)	FIE
(and nil ‘fie ‘foe)	NIL
(or (equal ‘abc ‘abc) ‘yes)	T

Задача 6.

Написать предикат, который принимает два числа-аргумента и возвращает T, если первое число не меньше второго.

Листинг 6: функция предикат для задачи №6.

```
(defun predicat(numOne numTwo)
        (cond
                ((>= numOne numTwo) T)))
```

Задача 7.

Какой из следующих двух вариантов предиката ошибочен и почему?

Вариант 1:

```
(defun pred1(x)
        (and (numberp x) (plusp x)))
```

Вариант 2:

```
(defun pred2(x)
        (and (plusp x) (numberp x)))
```

Второй вариант предиката ошибочен, так как сначала будет выполнен предикат plusp. plusp - это числовой предикат, поэтому при поступлении не числового аргумента произойдет ошибка.

Задача 8.

Решить задачу 4, используя для ее решения конструкции IF, COND, AND/OR.

Листинг 8.1: Применение конструкции IF для решения задачи №4.

```
(defun check-sequence(numOne numTwo numThree)
        (if (AND (>= numOne numTwo) (<= numOne numThree)) T Nil))
```

Листинг 8.2: Применение конструкции COND для решения задачи №4.

```
(defun check-sequence(numOne numTwo numThree)
        (cond
                ((AND (>= numOne numTwo) (<= numOne numThree)) T))
```

Задача 9.

Переписать функцию how-alike, приведенную в лекции и использующую COND, используя конструкции IF, AND/OR.

Листинг 9.1: функция how-alike с лекции.

```
(defun how-alike(x y)
        (cond ((or (= x y) (equal x y)) 'the_same)
                ((and (oddp x) (oddp y)) 'both_odd)
                ((and (evenp x) (evenp y)) 'both_even)
                (t 'difference)))
```

Листинг 9.2: функция how-alike с использованием if.

```
(defun how-alike(x y)
        (if (or (= x y) (equal x y)) 'the_same
                (if (and (oddp x) (oddp y)) 'both_odd
                        (if (and (evenp x) (evenp y)) 'both_even
                                'different))))
```

Контрольные вопросы

1. *Классификация функций*

Первый вариант классификации:

- Чистые функции - фиксированное количество аргументов, для определенного набора аргументов есть фиксированный результат;
- Функции формы (специальные функции) - функции, которые принимают произвольное количество аргументов или по разному обрабатывают результат ;
- Функции высшего порядка (Функционалы) - принимают или возвращают в качестве результата функцию.
- псевдофункции - создают эффект на экране;

Второй вариант классификации:

- селекторы
- конструкторы
- предикаты

2. *Работа функций and, or, if, cond*

and:
Логическая функция AND берет один или несколько аргументов. Она выполняет эти аргументы слева направо. Если она встречает аргумент, значение которого NIL, она возвращает NIL, не продолжая вычисления остальных. Если NIL аргументов не встретилось, то возвращается значение последнего аргумента.

or:
Логическая функция OR берет один или несколько аргументов. Она выполняет эти аргументы слева направо и возвращает значение первого аргумента, который не NIL. Если все аргументы OR имеют значение NIL, то OR возвращает NIL.

cond:
cond является основным средство организации разветвления вычислений.

Структура условного предложения :

(COND (< проверка-1 > < действие-1 >)

(< проверка-2 > < действие-2 >)

.....

(< проверка-n > < действие-n >))

Значение COND определяется следующим образом:

- Выражения **< проверка-i >**, выполняющие роль предикатов вычисляются последовательно, слева направо, до тех пор, пока не встретится выражение, значением которого не является **NIL**.

- Вычисляется результирующее выражение, соответствующее этому предикату, и полученное значение возвращается в качестве значения всего предложения **COND.**

- Если истинного значения нет, то значением **COND** будет **NIL.**

if:
cond является наиболее общим условным предложением. Обычно пользуются более простыми условными предложениями, например, if.

```
(IF <условие> <то форма> <иначе форма>)
```

3. Способы определения функций

Обычно функции определяются с помощью макроса defun.
Листинг 4.1: типовое использование макроса defun.

```
(defun name (parameter*)
    тело-функции*)
```

В качестве имени можно использовать любой символ, но обычно используются только буквы, цифры и знак минус. Рекомендуются избегать символа нижнего подчеркивания.

Список параметров функции определяет переменные, которые будут использоваться для хранения аргументов, переданных при вызове функции.

Тело defun состоит из произвольного числа s-выражений. При выполнении функции они будут выполнены по порядку, и будет возвращен результат последнего (в качестве результата работа всей функции).

Возможны ситуации, в которых определению новых функций при помощь defun является излишним. Для таких ситуаций в Lisp предусмотрена возможность создания анонимных функций при помощи выражения lambda.

Листинг 4.2: создание анонимной функции.

```
(lambda (parameters) body)
```