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
к лабораторной работе № 5
По курсу: «Функциональное и логическое программирование»

Студент Прохорова Л. А.

Группа ИУ7-63Б

Оценка (баллы) _____

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

Москва.
2021 г.

Цель работы: приобрести навыки работы с управляющими структурами в Lisp.
Задачи работы: изучить работу функций с произвольным количеством аргументов, функций разрушающих и неразрушающих структуру исходных аргументов.

Задание 1
Написать функцию, которая принимает целое число и возвращает первое четное число, не меньшее аргумента.

```
(defun f(num) (if (evenp num) num (+ num 1)))
```

(f 2) -> 2
(f 3) -> 3

Задание 2
Написать функцию, которая принимает число и возвращает число того же знака, но с модулем на 1 больше модуля аргумента.

```
(defun f(num) (if (< num 0) (- num 1) (+ num 1)))
```

Задание 3
Написать функцию, которая принимает два числа и возвращает список из этих чисел, расположенный по возрастанию.

С использованием list

```
(defun f_list(num1 num2) (if (<= num1 num2) (list num1 num2) (list num2 num1)))
```

(f_list 1 2) -> (1 2)
(f_list 4 3) -> (3 4)

С использованием cons

```
(defun f_cons(num1 num2)(if (<= num1 num2)(cons num1 (cons num2 Nil))(cons num2(cons num1 Nil))))
```

(f_list 1 2) -> (1 2)
(f_list 4 3) -> (3 4)

Задание 4
Написать функцию, которая принимает три числа и возвращает T только тогда, когда первое число расположено между вторым и третьим.

```
(defun is_mid(n1 n2 n3)(if (< n2 n1)(> n3 n1)(< n3 n1)))
```

(is_mid 1 2 3) -> Nil
(is_mid 2 1 3) -> T
(is_mid 2 3 1) -> T

Задание 5
Каков результат вычисления следующих выражений?

(and 'fee 'fie 'foe) -> foe
Так как аргумента Nil не встретилось, выводится значение последнего аргумента.

(or 'fee 'fie 'foe) -> fee
Функция or возвращает значение первого аргумента, который не Nil.

(and (equal 'abc 'abc) 'yes) -> yes
Так как (equal 'abc 'abc) -> T и Nil отсутствует, то выводится значение последнего аргумента.

(or nil 'fie 'foe) -> fie
Функция or возвращает значение первого аргумента, который не Nil.

(and nil 'fie 'foe)-> nil

(or (equal 'abc 'abc) 'yes) -> T
Так как (equal 'abc 'abc) -> T, а or возвращает первое значение, которое не Nil.

Задание 6
Написать предикат, который принимает два числа-аргумента и возвращает T, если первое число не меньше второго.

```
(defun p_n1(n1 n2)(>= n1 n2))
(p_n1 2 3) -> Nil
(p_n1 4 2) -> T
(p_n1 4 4) -> T
```

Задание 7
Какой из следующих двух вариантов предиката ошибочен и почему?

```
(defun pred1 (x)                    (defun pred2 (x)
(and (numberp x) (plusp x)))        (and (plusp x)(numberp x)))
```

Функция plusp принимает на вход тип real. Функция pred2 написана неверно, так как в случае, когда на вход функции pred2 в качестве аргумента будет подано не число, а , например, строка, произойдет ошибка в функции plusp. Если же передать в функцию pred1 в качестве аргумента строку, то сначала будет произведена проверка, что аргумент является числом, получится значение NIL и and вернет его в качестве результата.

Задание 8
Решить задачу 4 используя для её решения конструкции IF, COND, AND/OR.

Конструкция if:
```
(defun mid_if(n1 n2 n3)(if (< n2 n1)(> n3 n1)(< n3 n1)))
```

Конструкция cond:
```
(defun mid_cond(n1 n2 n3)
(cond
((< n2 n1) (> n3 n1))
((> n2 n1)(< n3 n1))))
```

Конструкция and/or:
```
(defun mid_ao(n1 n2 n3)(or (and (> n2 n1)(> n1 n3))(and (> n1 n2)(> n3 n1))))
```

Задание 9
Переписать функцию how-alike, приведенную в лекции и не использующую COND, используя конструкция IF, AND/OR.

```
(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)))

(defun my_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 'defference))))
```

Ответы на теоретические вопросы:
1.Классификация функций
Функции с точки зрения организации

- Чистые математические функции (фиксированное количество элементов и определён результат)
- Рекурсивные функции
- Специальные функции (произвольное количество аргументов или по разному обрабатывают свои аргументы)
- Псевдофункции (создают эффект на экране)
- Функции с вариантами значений из которых возвращается только одно значение

- Функции высших порядков (Используются для построения синтаксически управляемых программ. Это абстракция языка.Если описание одной функции синтаксически похоже на описание другой, то вместо двух похожих пишется одна)

Функции с точки зрения действий

- Конструкторы (Пример: cons, list)
- Селекторы (Пример: car, cdr)
- Предикаты (в математике это логические функции, для Lisp используется принцип: то что не Nil, является true) Пример: eq

2. Работа функций and, or, if, cond.

В обычных языках программирования существуют средства управления вычислительным процессом: организация разветвлений и циклов.
В Lisp для этих целей используются управляющие структуры - предложения.
Внешне предложения записываются как вызовы функций:
Первый элемент предложения - имя, остальные - аргументы.
В результате вычисления предложения получается значение. Отличие от вызова функции в использовании аргументов.

COND

```
(cond (test1 value1)
      (test1 value1)
          …
      (testN valueN)
```

В качестве аргументов **< test >** и **< value >** могут быть произвольные формы.

Значение COND определяется следующим образом:

- Выражения **< test-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.**

Обычно в качестве последнего условия пишется **t**, соответствующее ему выражение будет вычисляться в тех случаях, когда ни одно другое условие не выполняется.

IF

(IF test T_body F_body)
Вычисляется test, если он не Nill, то обрабатывается T_body, а F_body не обрабатывается. Если test Nill, то обрабатывается F_body.

AND

(and arg1 arg2 … argN)

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

(or arg1 arg2 … argN)

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.

3.Способы определения функций.
Для создания функции служит специальная встроенная функция DEFUN. Вызов этой функции требует трех аргументов:

- Первый аргумент должен атомом;
- Второй аргумент должен быть списком атомов;
- Третий аргумент может быть произвольной формой (S-выражением, имеющим значение)

Если вызов функции DEFUN завершился удачно, то в качестве результата возвращается атом, заданный первым параметром, а в системе появляется новая функция, требующая при вызове столько аргументов, сколько элементов

содержалось в списке формальных параметров. При вычислении этой новой функции значения аргументов будут вычисляться.

Например: (defun sum(arg1, arg2) (+ arg1 arg2)) .

Также функций можно определять через лямбда выражения. Лямбда выражение — это список, содержащий в себе слово lambda и список аргументов и следующие за ним тело функции, состоящее из 0 или более выражений.
Например: (lambda (x y) (+ x y)).
Вычисление лямбда функций происходит в два этапа: сначала вычисляются фактические параметры и связываются с формальными, а затем уже происходит выполнение функции. Плюс использования лямбда функции в том, что мы можем возвращать не только целые числа и так далее, но и функции.