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»

НАПРАВЛЕНИЕ ПОДГОТОВКИ «09.03.04 Программная инженерия»

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

Название: Работа интерпретатора Lisp

Дисциплина: Функциональное и логическое программирование

Студент	ИУ7-66Б		Т. А. Казаева
	Группа	Подпись, дата	И. О. Фамилия
Преподаватель			Н. Б. Толпинская
		Подпись, дата	И. О. Фамилия

Москва, 2022 г.

1. Теоретические вопросы

1. *Базис Lisp.*

   a) атомы и структуры (представляющиеся бинарными узлами);

   b) несколько базовых функций и функционалов: встроенные — примитивные функции (atom, eq, cons, car, cdr); специальные функции и функционалы (quote, cond, lambda, eval, apply, funcall).

2. *Классификация функций.*

   a) чистые (математические) функции: имеют фиксированное количество аргументов и в качестве возврата единственное значение;

   b) рекурсивные функции;

   c) специальные функции (формы): имеют произвольное количество аргументов, либо эти аргументы обрабатываются не все одинаково;

   d) псевдофункции: функции, эффект которых виден на внешних устройствах;

   e) функции с вариантными значениями, из которых выбирается одно;

   f) функции высших порядков (функционалы) используются для построения синтаксически-управляемых программ, в качестве одного из аргументов принимают описание функции.

3. *Способы создания функций*

Обычно функции определяются при помощи макроса DEFUN. В качестве имени может использоваться любой символ. Как правило, имена функций содержат только буквы, цифры и знак минус. Список параметров функции определяет переменные, которые будут использоваться для хранения аргументов, переданных при вызове функции. Тело DEFUN состоит из любого числа выражений Lisp.

4. *Работа функций Cond, if, and/or*

IF — (if test then-body else-body) является формой. Сначала вычисляется тестовое test-выражение. Если оно истинно, вычисляется then-выражение и возвращается его значение. В противном случае вычисляется else-выражение.

COND — имеет два преимущества: неограниченное количество условных переходов и неявное использование progn в каждом из них. Его имеет смысл использовать в случае, когда третий аргумент if – другое условное выражение. Каждый аргумент должен быть представлен списком, состоящим из условия и следующих за ним выражений, которые будут вычисляться в случае истинности этого условия. При вычислении вызова cond условия проверяются по очереди до тех пор, пока одно из них не окажет ся истинным. Далее вычисляются выражения, следующие за этим условием, и возвращается значение последнего из них. Если после выражения, оказавшегося истинным, нет других выражений, то возвращается его значение.

OR, AND — оба могут принимать любое количество аргументов, но вычисляют их до тех пор, пока не будет ясно, какое значение необходимо вернуть. Если все аргументы истинны (не nil), то and (or) вернет значение последнего.

2. Практические задания

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

```
(defun closest-even (x)
(if (evenp x)
  x
  (if (> x 0)
    (- x 1)
    (+ x 1))))
```

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

```
(defun greater (x)
  (if (< x 0)
    (- x 1)
    (+ x 1)))
```

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

```
(defun pair (x y)
  (if (< x y)
    (list x y)
    (list y x)))
```

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

```
(defun is-between (f s th)
  (if (or (and (> f s) (< f th)) (and (> f th) (< f s)))
      t
      nil))
```

5. *Каков результат вычисления следующих выражений?*

1) (**and** 'fee 'fie 'foe)

2) (**or** nil 'fie 'foe)

3) (**and** (**equal** 'abc 'abc)'yes)

4) (**or** 'fee 'fie 'foe)

5) (**and** nil 'fie 'foe)

6) (**or** (**equal** 'abc 'abc) 'yes)

Решения:

1) FOE
2) FIE
3) YES
4) FEE
5) NIL
6) T

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

```
(defun not-less (x y)
  (>= x y))
```

7. *Какой из следующих двух вариантов предиката ошибочен и почему?*

1)
```
(defun pred1 (x)
  (and (numberp x) (plusp x)))
```

2)
```
(defun pred2 (x)
  (and (plusp x) (numberp x)))
```

Решение.
Поскольку AND вычисляет аргументы до тех пор, пока не будет ясно, какой ответ надо вернуть, предикат 2 при первой проверке вернет NIL и завершит работу не вызывая plusp. Значит, ошибочен вариант №2.

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

```
(defun is-between-cond (lst)
  (cond ((and (> (first lst) (second lst)) (< (first lst) (third lst))) t)
        (t nil)))
```

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

```
(defun is-betweenp(f s th)
  (or (and (> f s) (< f th)) (and (> f th) (< f s))))
```

COND:

```
(defun is-between-cond(f s th)
  (cond ((> f s) (cond ((< f th))
                       (t nil)))
        ((> f th) t)))
```

IF:

```
(defun is-between-if (f s th)
  (if (> f s)
    (if (< f th)
      t
      (if (< f s)
        (if (> f th)
          t
          nil)
        nil))
    (if (> f th)
      t)))
```