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 №3

по курсу «Функциональное и Логическое программирование»

на тему: «Работа интерпретатора Lisp»

Студент ИУ7-63Б _____ Лагутин Д. В.

(Группа) (Подпись, дата) (Фамилия И. О.)

Преподаватель _____ Толпинская Н. Б.

(Подпись, дата) (Фамилия И. О.)

Москва, 2023 г.

Теоретические вопросы

1. Базис Lisp

Базис — минимальный набор инструментов и структкр данных языка, который позволяет реализовать любую поставленную задачи.

Базис языка представлен:

— атомами;

— структурами;

— функциями

`atom, eq, cons, car, cdr, cond, quote, lambda, eval, label.`

2. Классификация функций

— Чистые — не зависят от внешних, глобальных данных, не создают побочных эффектов.

— Формы:

  – могут иметь переменное количество параметров;
  – к арнументам может применяться особая обработка.

— Функционалы:

  – могут принимать функцию в качестве аргумента;
  – могут возвращать функцию.

Классификация базисных функций:

— селекторы;

— конструкторы;

— предикаты;

— функции сравнения.

3. Способы создание функций

Функия может быть определена двумя способами. С помощью λ-выражения `(lambda (`λ`-list) f)`, где λ-list — список формальных аргументов, а `f` - тело функции, или макро-определения `(defun name `λ`-выражение)`, где name — имя определяемой функции.

4. Работа функций cond, if, and/or

`(cond [(test [expression [...]]) [...]])`

Вычисляет выражения `test` до тех пор, пока одно из них не окажется истинным. Если не переданы какие-либо выражения `expression`, возвращается значение `test`, в противном случае выражения вычисляются по очереди и возвращается значение последнего. Если ни одно тестовое выражение не оказалось истинным или не указано вообще, возвращается `nil`.

`(if test then [else])`

Вычисляет значение `test`. Если оно истинно, вычисляет и возвращает значение выражения `then`, в противном случае вычисляет и возвращает значение `else` или возвращает `nil`, если оно отсутствует.

`(and [expression [...]])`

Вычисляет выражения по порядку. Если значение одного из них `nil`, дальнейшее вычисление прерывается и возвращается nil. Если все выражения истинны, возвращается значение последнего. Если выражения не заданы, возвращает `T`.

`(or [expression [...]])`

Вычисляет выражения друг за другом до тех пор, пока одно из значений не окажется истинным. В таком случае возвращается само значение, в противном случае – `nil`. Если выражения не заданы, возвращает `nil`.

1 Задание

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

```
1 (defun g-even (num) (+ num (mod num 2)))
```

2 Задание

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

```
1 (defun mod-sum (num)
2   (+ num (/ num (abs num))))
```

3 Задание

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

```
1 (defun sort-2 (num1 num2)
2   (cond ((> num1 num2) (list num2 num1))
3         (T (list num1 num2))))
```

4 Задание

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

```
1 (defun between (num num-l num-h)
2   (or (< num-l num num-h)
3       (> num-l num num-h)))
```

5 Задание

Каков результат вычисления следующих выражений?

```
(and 'fee 'fie 'foe)
; FOE

(or nil 'fie 'foe)
; FIE

(and (equal 'abc 'abc) 'yes)
; YES

(or 'fee 'fie 'foe)
; FEE

(and nil 'fie 'foe)
; NIL

(or (equal 'abc 'abc) 'yes)
; T
```

6 Задание

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

```
1 (defun ge (num1 num2)
2   (let ((dif (- num2 num1))) (zerop (+ dif (abs dif)))))
```

7 Задание

Какой из следующих двух вариантов предиката ошибочен и почему?

```
1 (defun pred1 (x)
2   (and (numberp x) (plusp x)))
3
4 (defun pred2 (x)
5   (and (plusp x) (numberp x)))
```

Неверным является реализация pred2, так как проверка того, что x - число, производится после проверки знака числа. Таким образом, pred2 может завершиться с ошибкой, что не является корректным поведением.

8 Задание

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

```
1  (defun between-if (num num-l num-h)
2    (if (< num-l num)
3        (< num num-h)
4        (if (/= num-l num) (> num num-h) nil)))
5
6  (defun between-cond (num num-l num-h)
7    (cond ((< num-l num num-h))
8          ((> num-l num num-h))))
9
10 (defun between-and-or (num num-l num-h)
11   (or (and (< num-l num num-h))
12       (and (> num-l num num-h))))
```

9 Задание

Переписать функцию how-alike, приведенную в лекции и использующую COND, используя только конструкции 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 how-alike-if (x y)
  (if (/= x y)
      (if (not (equal x y))
          (if (oddp x)
              (if (oddp y) 'both_odd 'difference)
              (if (evenp y) 'both_even 'difference))
          'both_same)
      'both_same))

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