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)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04 Программная инженерия**

О Т Ч Е Т

по лабораторной работе № 5

Дисциплина: Функциональное и логическое программирование

Студент ИУ7-62Б _____ Е.В. Брянская
(Группа) (Подпись, дата) (И.О. Фамилия)

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

Москва, 2021

Лабораторная работа №5.

№3 : 1-9

(№1) Написать ф-цию, которая принимает целое число и возвращает первое четное число, не меньшее аргумента.

```
(defun f (num)
    (cond ((evenp num) num)
          (t (+ num 1))
    )
)
```

(№2) Написать ф-цию, которая принимает число того же знака, но с модулем на 1 больше модуля аргумента.

```
(defun f (num)
    (cond ((plusp num) (+ num 1))
          ((minusp num) (- num 1))
          (t "0 - не имеет знак")
    )
)
```

(№3) Написать ф-цию, которая принимает два числа и возвращает список из этих чисел, расположенных по возрастанию.

```
(defun f (arg1 arg2)
    (cond ((> arg1 arg2) (cons arg2 (cons arg1 nil)))
          (t (cons arg1 (cons arg2 nil)))
    )
)
```

(14) Написать ф-цию, которая принимает три числа и возвращает т только тогда, когда первое число расположено между вторым и третьим.

```
(defun f (arg1 arg2 arg3)
   (cond ((or (> arg2 arg1 arg3) (< arg2 arg1 arg3)) t))
)
```

(15) Каков результат вычисления следующих выражений?

```
(and 'fee 'fie 'foe) = foe
(or 'fee 'fie 'foe) = fee

(or nil 'fie 'foe) = fie
(and nil 'fie 'foe) = nil

(and (equal 'abc 'abc) 'yes) = yes
(or (equal 'abc 'abc) 'yes) = t
```

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

```
(defun fp (a1 a2)
)    (>= a1 a2)
```

(2) какой из следующих двух вариантов предиката ошибочен и почему?

1. (define pred1 (x)
   (and (numberp x) (plusp x)))
2. (define pred2 (x)
   (and (plusp x) (numberp x)))

Предикат numberp проверяет является ли аргумент числом

Предикат plusp проверяет является ли ~~аргумент~~ число положительным

Ошибочен предикат 2, так как если на вход будет передано не число, то первая проверка укажет на ошибку ("not a number") и при таком порядке условий or сработает. Так как, если аргумент - число, то проверка на положительность уже предполагает выполнение условия, что подано число, если же аргумент не число, то or условие не проверится, т.к. and завершит работу раньше

(NB) решить задачу 4, используя для ее решения конструкции IF, cond, AND/OR.

(IF)
```
(define f (arg1 arg2 arg3)
    (if (> arg2 arg1 arg3)
        T
        (< arg2 arg1 arg3)
    )
)
```

(cond)
```
(defun f (arg1 arg2 arg3)
    (cond ( (> arg2 arg1 arg3) t)
          ( (< arg2 arg1 arg3) t)
    )
)
```

(AND/OR)
```
(defun f (arg1 arg2 arg3)
    (or (> arg2 arg1 arg3)
        (< arg2 arg1 arg3)
    )
)
```

(№9) Переписать функцию how-alike, приведенную в лекции и использующую cond, используя 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)))
```

(IF)
```
(defun how-alike (x y)
  (if (= x y)
      'the same
      (if (equal x y)
          'the same
          (if (oddp x)
              (if (oddp y)
                  'both_odd
                  'difference
              (if (evenp y)
                  'both_even
                  'difference)))))
)
```

```
(define (how-alike (x y)
  (cond ((and (= x y) 'the-same)
        (and (equal x y) 'the-same)
        (and (oddp x) (oddp y) 'both-odd)
        (and (evenp x) (evenp y) 'both-even)
        'difference
  )
)
```

Вопрос:

① Классификация функций

Функции:

1. числовые математические
2. рекурсивные
3. спец. функции, формы
4. псевдофункции
5. Ф-ции с вариантами значений, но выбирается единственный результат
6. Ф-ции внешних эффектов

Функции:

1. селекторы (car, cdr)
2. конструкторы (cons, list)
3. предикаты (atom, null, eq, consp, listp, numberp...)

② Работа функций and, or, if, cond

and

(and arg1 arg2 ... argN)

Вычисляет свои аргументы до тех пор, пока не будет очевиден результат. Если первый аргумент истина, то осуществляется переход к следующему аргументу и т.д. Если все аргументы принимают истинное значение, то на экран возвращается последнее выражение (его значение).
Если очередной аргумент равен nil, то всё S-выражение nil и прекращается обработка аргументов.

or

(or arg1 arg2 ... argN)

Также вычисляет свои аргументы до тех пор, пока не будет очевиден результат.
Как только аргумент принимает значение T, то работа прекращается и остальные аргументы не вычисляются, возвращается значение этого истинного выражения.

if

(if test T_body F_body)

Сначала вычисляется test-выражение, если оно T то вычисляется T_body-выражение и возвращается его значение; если же test-выражение nil, то F_body и возвращается уже его значение.

cond

```
(cond (test1 value1)
      (test2 value2)
      ...
      (testN valueN)
)
```

Аргументами является список из двух элементов. test - S-выражение, которое вычисляется и его значение воспринимается как логическое (либо T, либо nil).

или оно T, то вычисляется второй элемент из этого списка, его значение возвращается и работа завершается.

Если nil, то вычисляется test2 и т.д

Если ни одно условие не выполнено, то cond вернет nil.

③ Способы определения функции

В Lisp можно определить функцию без имени с помощью λ-выражений.

(lambda (X1 X2 ... Xn) fn) → тело функции

(X1 X2 ... Xn) — λ-список форм. парам.

Можно определить и именованные ф-ции с помощью defun

(defun имя λ-список тело)

имя → символьный атом

λ-список → список форм. парам.

тело → определение ф-ции

Применить ф-цию:

(funcall #' имя или λ-выражение факт.пар.)

(apply #' имя или λ-выражение (факт.пар.))