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 №5

Название: Использование функционалов

Дисциплина: Функциональное и логическое программирование

Студент ИУ7-66Б _____ А.Д. Ковель
Группа Подпись, дата И. О. Фамилия

Преподаватель _____ Н. Б. Толпинская

Преподаватель _____ Ю. В. Строганов
Подпись, дата И. О. Фамилия

Москва, 2023 г.

1 Практические задания

1. *Напишите функцию, которая уменьшает на 10 все числа из списка-аргумента этой функции, проходя по верхнему уровню списковых ячеек. (* Список смешанный структурированный)*

```
(defun minus-ten-list (lst) (
  mapcar #'(
        lambda (elem) (
          cond
            ((numberp elem) (- elem 10))
            (T elem)
        )
      ) lst
  )
)
```

2. *Написать функцию которая получает как аргумент список чисел, а возвращает список квадратов этих чисел в том же порядке.*

```
(defun square-lst (lst) (
  mapcar #'(
        lambda (elem) (
          cond
            ((numberp elem) (* elem elem))
            (T elem)
        )
      ) lst
  )
)
```

3. *Напишите функцию, которая умножает на заданное число-аргумент все числа из заданного списка-аргумента, когда а) все элементы списка — числа, б) элементы списка – любые объекты.*

```

(defun multiply-lst-numbers (lst x) (
  mapcar #'(
        lambda (elem) (
          * elem x
```

```
            )
          ) lst
      )
    )


    (defun multiply-lst (lst x) (
      mapcar #'(
            lambda (elem) (
              cond
                ((numberp elem) (* elem x))
                (T elem)
            )
          ) lst
      )
    )
```

4. *Написать функцию, которая по своему списку-аргументу lst определяет является ли он палиндромом (то есть равны ли lst и (reverse lst)), для одноуровнего смешанного списка.*

```
    (defun get-last (lst)
      (
        nbutlast lst 1
      )
    )
```

5. *Используя функционалы, написать предикат set-equal, который возвращает t, если два его множества-аргумента (одноуровневые списки) содержат одни и те же элементы, порядок которых не имеет значения.*

```
    (defun swap-first-last (lst)
      (
        nconc
        (last lst)
        (reverse
          (cdr
            (reverse (cdr lst)))
        )
        (list (car lst))
      )
```

```
11      )
```

6. *Напишите функцию, select-between, которая из списка-аргумента, содержащего только числа, выбирает только те, которые расположены между двумя указанными числами - границами-аргументами и возвращает их в виде списка (упорядоченного по возрастанию (+ 2 балла)).*

```
1       (defvar first_player)
2       (defvar second_player)
3
4
5       (defun bones_throw ()
6
7         (print "Enter␣first␣bone:␣")
8         (setq bone1 (read))
9         (print "Enter␣second␣bone:␣")
10        (setq bone2 (read))
11        (setq ret (list bone1 bone2))
12        ret
13
14      )
```

7. *Написать функцию, вычисляющую декартово произведение двух своих спискоаргументов. (Напомним, что A x B это множество всевозможных пар (a b), где a принадлежит A, принадлежит B.)*

```
1       (defun get-without-last-reverse (lst)
2         (reverse (cdr (reverse lst)))
3       )
4
5
6
7       (defun st_check (lst)
8         (cond
9           (
10            (> (length lst) 1)
11            (and
12              (eq (car lst) (car (reverse lst)))
13              (st_check (cdr (get-without-last-reverse lst)))
14            )
15          )
```

```
16          (T T)
17        )
18      )
19
20
21      (defun palindrom_check (lst)
22        (st_check lst)
23      )
```

8. *Почему так реализовано reduce, в чем причина? (reduce #'+ ()) -> 0 (reduce #'* ()) -> 1*

```
1       (defun countries_capitals (lst name)
2         (
3           cond
4           (
5             (assoc name lst)
6             (cdr (assoc name lst))
7           )
8           (
9             (rassoc name lst)
10            (car (rassoc name lst))
11          )
12          (T Nil)
13        )
14      )
```

9. ** Пусть list-of-list список, состоящий из списков. Написать функцию, которая вычисляет сумму длин всех элементов list-of-list (количество атомов), т.е. например для аргумента ((1 2) (3 4)) -> 4.*

```
1       (defun mult_el_a (n lst)
2         (
3           cond
4           (
5             (and
6               (and
7                 (numberp (car lst))
8                 (and (numberp (cadr lst)) (numberp (caddr lst)))
9               )
10              (numberp n)
11            )
```

```
12            (* (car lst) n)
13          )
14          (T Nil)
15        )
16      )
```