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

по лабораторной работе №5

Название «Использование управляющих структур, работа со списками»

Дисциплина «Функциональное и логическое программирование»

Студент ИУ7-65Б _____ Бугаенко А.П.
(подпись, дата) (Фамилия И.О.)

Преподователь _____ Толпинская Н.Б.
(подпись, дата) (Фамилия И.О.)

Москва, 2022

1 Цели и задачи работы

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

2 Теоретические вопросы

2.1 Структуроразрушающие и не разрушающие структуру списка функции

Функции можно разбить на две группы - разрушающие структуру и не разрушающие структуру. Для работы со списком его необходимо создать, получить доступ и модифицировать. Функции, разрушающие структуру - изменяют структуру списка. Функции не разрушающую структуру - производят какие-то операции без изменения поданного на вход списка.

В функции, разрушающие структуру списка входят - nconc, nreverse В функции, не разрущающие структуру списка входят - append, reverse, list, cons

2.2 Отличие в работе и результат работы функций cons, list, append, nconc

При работе функции cons создаётся бинарный узел, car которого указывает на первый аргумент, а cdr на второй. В результате работы создаётся новый объект и при этом не разрушаются списки, поданные на вход.

При работе функции list создаётся новый список из поданных на вход элементом, при этом сами элементы не меняются.

При работе функции append создаётся копия списка, в конец которой добавляется новый элемент. При этом старый список остаётся без изменений.

При работе функции nconc новый список создаётся путём присвоения cdr указателям концов списков начал следуюещго списка. В результате разрушается структура списка, поскольку при попытке получить доступ к старому списку будет выводится новый список начиная с аргумента, с которым был связан символ.

3 Практические задания

3.1 Задание 1

Написать функцию, которая по своему списку-аргументу lst определяет является ли он палиндромом (то есть равны ли lst и (reverse lst)).

```
(defun my_reverse_rec (lst new_lst)
  (if (not (eql (cdr lst) nil))
      (my_reverse_rec (cdr lst) (cons (car lst) new_lst))
      (cons (car lst) new_lst)))

(defun my_reverse (lst)
  (my_reverse_rec lst nil))

(defun cmp_list_rec (lst_1 lst_2)
  (if (eql (car lst_1) (car lst_2))
      (if (not (or (eql (cdr lst_1) nil) (eql (cdr lst_2) nil)))
          (cmp_list_rec (cdr lst_1) (cdr lst_2))
          (if (and (eql (cdr lst_1) nil) (eql (cdr lst_2) nil))
              t
              nil))
      nil))

(defun check_palindrom (lst)
  (cmp_list_rec lst (my_reverse lst)))
```

3.2 Задание 2

Написать предикат set-equal, который возвращает t, если два его множества-аргумента содержат одни и те же элементы, порядок которых не имеет значения.

```
(defun in-list (lst elem)
  (if (eql (car lst) elem)
      t
      (if (eql (cdr lst) nil)
          nil
          (in-list (cdr lst) elem)
          )
      )
  )

(defun set-equal-rec (set_1 set_2)
  (if (in-list set_1 (car set_2))
      (if (not (eql (cdr set_2) nil))
          (set-equal-rec set_1 (cdr set_2))
          t
          )
```

```
      nil
      )
  )

(defun set-equal (set_1 set_2)
  (and (set-equal-rec set_1 set_2) (set-equal-rec set_2 set_1))
  )
```

3.3 Задание 3

Напишите свои необходимые функции, которые обрабатывают таблицу из 4-х точечных пар: (страна . столица), и возвращают по стране - столицу, а по столице — страну.

3.4 Задание 4

3.5 Задание 5

3.6 Задание 6

3.7 Задание 7

3.8 Задание 8

3.9 Задание 9