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 №7 по курсу «Функциональное и логическое программирование»

Тема Модификация списков

Студент Кононенко С.С.

Группа ИУ7-63Б

Оценка (баллы) _____

Преподаватели Толпинская Н.Б., Строганов Ю.В.

Москва — 2021 г.

Задание 1

Постановка задачи. Написать функцию, которая по своему списку-аргументу `lst` определяет, ялвяется ли он палиндромом.

Решение.

Листинг 1 – Решение задания 1

```
(defun palindromep (lst)
    (equal lst (reverse lst)))
```

Задание 2

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

Решение.

Листинг 2 – Решение задания 2

```
(defun set-equal-p (flst slst)
    (and (subsetp flst slst) (subsetp slst flst)))
```

Задание 3

Постановка задачи. Напишите необходимые функции, которые обрабатывают таблицу из точечных пар: `(страна . столица)`, и возвращают по стране – столицу, а по столице – страну.

Решение.

Листинг 3 – Решение задания 3

```
(defun make-country (tbl cap)
    (cond ((null tbl) Nil)
    ((equal (cdar tbl) cap) (caar tbl))
    (T (make-country (cdr tbl) cap))))
(defun make-capital (tbl ctr)
```

```
    (cond ((null tbl) Nil)
    ((equal (caar tbl) ctr) (cdar tbl))
    (T (make-capital (cdr tbl) ctr))))
```

Задание 4

Постановка задачи. Напишите функцию `swap-first-last`, которая переставляет в списке-аргументе первый и последний элементы.

Решение.

Листинг 4 – Решение задания 4

```
(defun make-swap-first-last (lst)
    (let ((fel (car lst))
        (lel (last lst)))
    (setf (car lst) (car lel))
    (setf (car lel) fel)
    lst))
```

Задание 5

Постановка задачи. Напишите функцию `swap-two-elemets`, которая переставляет в списке-аргументе два указанных своими порядковыми номерами элемента в этом списке.

Решение.

Листинг 5 – Решение задания 5

```
(defun make-swap-two-els (lst i j)
    (let ((len (length lst))
        (lstcp (copy-list lst)))
    (and (< i len) (< j len)
    (let ((fel (nth i lst))
        (lel (nth j lst)))
    (setf (nth i lstcp) lel)
    (setf (nth j lstcp) fel)
    lstcp))))
```

Задание 6

Постановка задачи. Напишите две функции, `swap-to-left` и `swap-to-right`, которые производят круговую перестановку в списке аргументе влево и вправо, соответственно.

Решение.

Листинг 6 – Решение задания 6

```
(defun make-swap-to-left (lst)
    (and lst (append (cdr lst) (cons (car lst) Nil))))
(defun make-swap-to-right (lst)
    (and lst (append (last lst) (nreverse (cdr (reverse lst))))))
```

Ответы на контрольные вопросы

Вопрос 1. Способы определения функций.

Ответ. Функцию можно задать через функцию `defun` или `lambda`.

`(defun имя_функции (список_аргументов) тело_функции)`

Например, `(defun sum (x y) (+ x y))`

Вызов функции: `(sum 2 3) -> 5`

`(lambda (список_аргументов) тело_функции)`

Например, `(lambda (x y) (+ x y))`

Вызов функции: `(lambda (x y) (+ x y) 2 3) -> 5`

Вопрос 2. Варианты и методы модификации элементов списка.

Ответ.

Не разрушающие структуру функции

Данные функции не меняют сам объект-аргумент, а создают копию.

Функция `append`

Объединяет списки. Это форма, можно передать больше 2 аргументов. Создает копию для всех аргументов, кроме последнего.

Пример: `(append '(1 2) '(3 4))` — `(1 2 3 4)`.

Функция `reverse`

Возвращает копию исходного списка, элементы которого переставлены в обратном порядке. **В целях эффективности работает только на**

верхнем уровне.

Пример: `(reverse '(1 2 3 4))` — `(4 3 2 1)`.

Функция `last`

Проход по верхнему уровню и возврат последней списковой ячейки.

Пример: `(last '(1 2 3 4))` — `(4)`.

Функция `nth`

Возврат указателя от n-ной списковой ячейки, нумерация с нуля.

Пример: `(nth 1 '(1 2 3 4))` — `2`.

Функция `nthcdr`

Возврат n-ого хвоста.

Пример: `(nthcdr 1 '(1 2 3 4))` — `(2 3 4)`.

Функция `length`

Возврат длины списка (**только по верхнему уровню**).

Пример: `(length '(1 2 (3 4)))` — `3`.

Функция `remove`

Модифицирует, но работает с копией, поэтому не разрушает. Данная функция удаляет элемент по значению (Часто разрушающая аналогичная функция называется `delete`). По умолчанию используется `eql` для сравнения на равенство, но можно передать другую функцию через ключевой параметр `:test`.

Пример: `(remove 3 '(1 2 3))` — `(1 2)`;

Функция `rplaca`

Переставляет `car`-указатель на 2 элемент-аргумент (S-выражение).

Пример: `(rplaca '(1 2 3) 3)` — `(3 2 3)`.

Функция `rplacd`

Переставляет `cdr`-указатель на 2 элемент-аргумент (S-выражение).

Пример: `(rplacd '(1 2 3) '(4 5))` — `(1 4 5)`.

Функция `subst`

Заменяет все элементы списка, которые равны 2 переданному элементу-аргументу на другой 1 элемент-аргумент. *По умолчанию для сравнения используется функция* `eql`.

Пример: `(subst 2 1 '(1 2 1 3))` — `(2 2 2 3)`.

Структуроразрушающие функции

Данные функции меняют сам объект-аргумент, невозможно вернуться к исходному списку. Чаще всего такие функции начинаются с префикса `n-`.

Функция `nconc`

Работает аналогично `append`, только не копирует свои аргументы, а разрушает структуру.

Функция `nreverse`

Работает аналогично `reverse`, но не создает копии.

Функция `nsubst`

Работае аналогично функции `nsubst`, но не создает копии.