|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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»

**Лабораторная работа № 10**

|  |  |
| --- | --- |
| **По дисциплине**: Функциональное и логическое программирование  **Тема: «**Использование управляющих структур, работа со списками**»**  **Студент** Козлов М.А.  **Группа** ИУ7-65  **Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**  **Преподаватель** Толпинская Н.Б., Строганов Ю.В. |  |

Москва.

2021 г.

1. Написать рекурсивную версию (с именем reg-add) вычисления суммы чисел заданного списка.

Например: (reg-add (2 4 6)) → 12

|  |
| --- |
| (defun rec\_add (lst)  (reduce #'(lambda (res x) ( + res x)) lst))  (defun rec\_add\_tail (lst rst)  (if (null lst) rst  (rec\_add\_tail (cdr lst) (+ (car lst) rst))  )  ) |

2. Написать рекурсивную версию с именем recnth функции nth.

|  |
| --- |
| (defun rec\_nth (lst k)  (if (< k 1) lst  (rec\_nth (cdr lst) (- k 1))  )  ) |

3. Написать рекурсивную функцию alloddr, которая возвращает t когда все элементы списка нечетные.

|  |
| --- |
| (defun alloddp (lst)  (or (null lst)  (and  (oddp (car lst))  (alloddp (cdr lst))  )  )  ) |

4. Написать рекурсивную функцию, относящуюся к хвостовой рекурсии с одним тестом завершения, которая возвращает последний элемент списка - аргументы.

|  |
| --- |
| (defun last\_m (lst)  (if (null (cdr lst)) lst (last\_m (cdr lst)))  ) |

5. Написать рекурсивную функцию, относящуюся к дополняемой рекурсии с одним тестом завершения, которая вычисляет сумму всех чисел от 0 до n-ого аргумента функции.

|  |
| --- |
| (defun sum (lst)  (if (null lst) 0 (+ (car lst) (sum (cdr lst))))  )  (defun sum\_a (lst n)  (cond ((null lst) 0)  ((= 0 n) (if (>= (car lst) 0)  (+ (car lst) (sum\_a (cdr lst) n))  0  )  )  (t (sum\_a (cdr lst) (- n 1)))  )  )  (defun sum\_b (lst n m d)  (cond ((null lst) 0)  ((= 0 n)  (if (>= m n)  (+ (car lst) (sum\_b (rec\_nth lst d) n (- m d) d))  0  )  )  (t (sum\_b (cdr lst) (- n 1) m d))  )  ) |

6. Написать рекурсивную функцию, которая возвращает последнее нечетное число из числового списка, возможно создавая некоторые вспомогательные функции.

|  |
| --- |
| (defun lastodd\_help(lst res)  ( cond ((null lst) res)  ((oddp (car lst)) (lastodd\_help (cdr lst) (car lst)))  (t (lastodd\_help (cdr lst) res))  )  )  (defun lastodd(lst)  (lastodd\_help lst nil)) |

7. Используя cons-дополняемую рекурсию с одним тестом завершения, написать функцию которая получает как аргумент список чисел, а возвращает список квадратов этих чисел в том же порядке.

|  |
| --- |
| (defun squaring (lst)  (cond ((null lst) lst)  (t (cons (\* (car lst) (car lst)) (squaring (cdr lst))))  )  ) |

8. Написать функцию с именем select-odd, которая из заданного списка выбирает все нечетные числа.

|  |
| --- |
| (defun select (lst predicate)  (mapcan #'(lambda (x)  (if (funcall predicate x) (cons x nil))  ) lst  )  ) |

(select '(1 2 3 5 6) #'(lambda (x) (oddp x))) => (1 3 5)

9. Создать и обработать смешанный структурированный список с информацией:

ФИО, зарплата, возраст, категория(квалификация). Изменить зарплату, в зависимости от заданного условия, и подсчитать суммарную зарплату. Использовать композиции функций.

|  |
| --- |
| (defun create-table (lst lastname firstname middlename salary age rank)  (cons (list lastname firstname middlename salary age rank) lst)  )  (defun update-salary-all(lst new\_salary)  (mapcar  #'(lambda (x)  (setf (cadddr x) new\_salary)  ) lst  )  )  (defun update-salary-person (lst new\_salary pred)  (mapcar  #'(lambda (x)  (if (funcall pred x)  (setf (caddr x) new\_salary)  )  ) lst  )  )  (defun count-salary (lst)  (cond  ((null lst) 0)  (t (+ (car (cdddar lst)) (count-salary (cdr lst))) )  )  ) |

(update-salary-all (create-table '((A B C 4000 20 2)) 'D 'E 'M 5000 22 1) 3000) =>

(3000 3000)

(update-salary-person (create-table '((A B C 4000 20 2)) 'D 'E 'M 5000 22 1) 3000 #'(lambda (x) (and (= 2 (nth 5 x) )))) => (NIL 3000)

(count-salary (create-table '((A B C 4000 20 2)) 'D 'E 'M 5500 22 1)) => 9500