http://it.kgsu.ru/Lisp/lisp0056.html

```
;gnu clisp 2.49

;Из списка получить список только из символов
(defun extract_symb(lst)
   (cond ((null lst) nil)
         ((symbolp (car lst)) (cons (car lst) (extract_symb (cdr lst))))
         (T (extract_symb (cdr lst)))))

(setf a '(34 5 g d hsd 5 po q))

;(print (extract_symb a))


;Многоуровневый символ, получить вернуть первый символ
(defun first_num(lst)
   (cond ((numberp lst) lst)
         ((atom lst) nil)
         (t (or (first_num (car lst)) (first_num (cdr lst))))))

(setf b '(() sfj e p ui (() (r c g (3)))) )

;(print (first_num b))


;Длина списка
(defun my_length(lst)
   (cond ((null lst) 0)
         (t (+ 1 (my_length (cdr lst))))))

;(print (my_length a))


;Список в один уровень
(defun into_one(lst rst)
   (cond ((null lst) rst)
         ((atom lst) (cons lst rst))
         (t (into_one (car lst) (into_one (cdr lst) rst)))))
```

```
;(print (into_one b nil))


;Сортировка вставкой
(defun insert_help(x lst)
   (cond ((null lst) (list x))
         ((<= x (car lst)) (cons x lst))
         (t (cons (car lst) (insert_help x (cdr lst)))) ))

(defun sort_help(lst1 lst2)
   (cond ((null lst1) lst2)
         (t (sort_help (cdr lst1) (insert_help (car lst1) lst2))) ))

(defun sort_ins(lst)
   (sort_help lst nil))

;(setf i '(5 3 7 2 8 1 0))
;(print (sort_ins i))


;Сортировка слиянием
;(defun merge(lst1 lst2 res)
;    (cond ((and (null lst1)(null lst2)) res)
;          ((null lst1) (cons lst2 res))
;          ((null lst2) (cons lst1 res))
;          (T (cond ((<= (car lst1)(car lst2)) (merge (cdr lst1) lst2 (cons (car lst1) res)))
;                   (t (merge lst1 (cdr lst2) (cons (car lst2) res))) ))))

;(defun merge(lst1 lst2 res)
;    (cond ((and (null lst1)(null lst2)) res)
;          ((null lst1) (cons res lst2 ))
;          ((null lst2) (cons res lst1 ))
;          (T (cond ((<= (car lst1)(car lst2)) (merge (cdr lst1) lst2 (cons (car lst1) res)))
;                   (t (merge lst1 (cdr lst2) (cons (car lst2) res))) ))))

(setf first '(1 2 3 3 6 7))
(setf second '(1 3 4 4 5 8 9))

;(print (merge first second nil))

(defun lst_rhalf_help(lst n curr)
```

```
   (cond ((eq curr n) lst)
         (t (lst_rhalf_help (cdr lst) n (+ curr 1) )) ))

(defun lst_rhalf(lst n)
   (lst_rhalf_help lst n 0))


(defun lst_lhalf(lst n)
   (cond ((eq n 0) nil)
         (t (cons (car lst) (lst_lhalf (cdr lst) (- n 1))) ) ))


(setf f '(4 2 6 6 5 1 8 6 2))

;2. посчитать среднее значение чисел в списке, которые на нечетных позициях.
;Когда используешь функционалы, рекомендуют делать с редусом и
;скакойто его настройкой инит-валю-чего-то-там, чтоб было
(setf ff '(1 3 2 7 5))
(defun middle_value (table)
   (setf pos 0)
   (reduce (lambda (x y) (if (evenp (cdr x)) (cons (cons (+ y (caar x)) ( + 1 (cdar x)))
(+ 1 (cdr x)))
                                          (cons (cons (caar x) ( cdar x)) ( + 1 (cdr x)))
                          )) table :initial-value (cons (cons 0 0) 0))
)
(print (middle_value ff))
; B1. В структурированном (сложном списке, в котором другие списки) один
заданный элемент заменить на другой. Например 1 на 6.


(setf wwww '(4 2 (6 (6)) 5 1 8 6 2))
;Рекурсивно
(defun lst_change(lst elem chelem)
   (cond ((null lst) ())
         ((equalp elem (car lst)) (cons chelem (lst_change (cdr lst) elem chelem)))
         ((listp (car lst)) (cons (lst_change (car lst) elem chelem) (lst_change (cdr lst)
elem chelem)))
         (t (cons (car lst) (lst_change (cdr lst) elem chelem)))))

;Через функционалы...
(defun lst_change2(lst elem chelem)
   (mapcar (lambda(x) (cond ((equalp x elem) chelem)
```

```
                        ((listp x) (lst_change2 x elem chelem) )
                         (T x))) lst ))

;(print (lst_change2 wwww 6 666))

;(print (lst_lhalf f 4))
;(print (lst_rhalf f 4))



;(defun merge_sort(lst)
;    (cond ((<= (length lst) 1) lst)
;          (t (merge (merge_sort ) (merge_sort)) )
;


;N1 (a)
;A и B числовые множества, полученные из двух смешанных (я так понимаю неотсортированных) одноуровневых списков. Найти A\B
;A \ B - числа из A, которые не содержатся в B.
;Решение: Сортируем, далее делаем процедуру, похожую на слияние, только уже с учетом условия.
;Время работы: O(сортировки двух множеств) + O(|A| + |B|)

;Код на python для лучшего понимания
;def difference(A, B):
;    if len(B)==0:
;        return A # return the leftover list
;    if len(A)==0:
;        return B # return the leftover list
;    if A[0] < B[0]:
;        return [A[0]] + difference(A[1:], B)
;    elif A[0] == B[0]:
;        return difference(A[1:], B[1:])  # omit the common element
;    else:
;        return difference(A, B[1:])

;Рекурсивно
(defun diff(a b)
   (cond ((null b) a)
         ((< (car a)(car b)) (cons (car a) (diff (cdr a) b) ))
         ((> (car a)(car b)) (diff a (cdr b)) )
         (t (diff (cdr a) (cdr b))) ))
```

```
(defun set_diff(a b)
   (diff2 (sort_ins a) (sort_ins b)))

;Функционалы
(defun diff2(lst1 lst2)
   (remove-if (lambda (x) (member x lst2 :test #'equalp)) lst1)
)



;(setf a '(3 9 8 7 6 5))
;(setf b '(5 9 1 6 2))
;(print (set_diff a b ))



;N1 (b)
;Дан структурированный список (я так понимаю многоуровневый). Реализовать
удаление
;из списка атома, списка

;Рекурсивно
;Как я офигел, когда оно заработало)))
(defun lst_remove(lst elem)
   (cond ((null lst) ())
         ((equal elem (car lst)) (lst_remove (cdr lst) elem))
         ((listp (car lst)) (cons (lst_remove (car lst) elem) (lst_remove (cdr lst) elem)))
         (t (cons (car lst) (lst_remove (cdr lst) elem)))))

;Через функционалы...
(defun lst_remove2(lst elem)
   (mapcar (lambda(x) (if (listp x) (lst_remove2 x elem) x)) (remove elem lst :test
#'equalp )))

;(setf lst '(3 2 5 (3 4 ((5) (2 7 3)((5 (2 7 3))))) 6 (2 7 3) 7 3))
;(print (lst_remove2 lst  '(2 7 3)))



;N2 (a)
```

```
;Реализовать добавление в ассоциативный список (элементы списка представлены в виде (ключ . значение))
;нескольких точечных пар, заданных списком ключей и списком значений.
;Насколько я понял ее конченную формулировку, на вход идет ассоциативный список и еще два (список ключей, список значений).
;В словаре не может быть пар с одинаковыми ключами (Это мы не проверяем)

(defun lists_to_dict(key_lst val_lst)
   (cond ((null key_lst) nil)
         (t (cons (cons (car key_lst) (car val_lst)) (lists_to_dict (cdr key_lst) (cdr val_lst))) ) ))


(defun list_append (l1 l2)
  (cond ((null l1) l2)
        (t (cons (car l1) (list_append (cdr l1) l2))) ))

(defun dict_insert(dict key_lst val_lst)
   (list_append dict (lists_to_dict key_lst val_lst)) )

; Функционалы
(defun add_items2 (lst key_list val_list)
          (nconc lst (mapcar #'(lambda (x y) (cons x y)) key_list val_list))
)

(setf a '((4 . 2) (5 . 6)) )
(setf b '(2 3 6))
(setf c '(4 5 6))
;(print (add_items2 a b c))



;N2 (b)
;Дан смешанный структурированный список. Получить по возрастанию список из числовых
;элементов исходного списка, входящих в заданное в виде одноуровнего смешанного списка множество.
;Ну кааак так можно формулировать задания, вот что тут надо делать????

;Гипотеза №1. Получить все числа из многоуровнего списка, отсортировать это дело.
```

```
;Гипотеза №2. Нам задается помимо многоуровнего еще и одноуровневый, в котором, содержаться
;определенные числа. Мы получаем все числа из многоуровнего и результатом будут числа которые содержатся
;как в заданном, так и полученных списках (В отсортированном порядке).

;Алгоритм решения задачи второй гипотезы. Получаем числа из многоуровнего, сортируем эти числа. Сортируем
;заданный список, находим пересечение этих двух множеств.


(defun intersect(lst1 lst2)
   (cond ((or (null lst1) (null lst2)) nil)
         ((< (car lst1) (car lst2)) (intersect (cdr lst1) lst2))
         ((> (car lst1) (car lst2)) (intersect lst1 (cdr lst2)))
         (t (cons (car lst1) (intersect (cdr lst1) (cdr lst2)))) ))


(defun solution(lst1 lst2)
   (intersect (sort_ins (into_one lst1 nil)) (sort_ins lst2)) )


;(setf lst '(3 2 5 ( 4 ( ( 7 )(((9 11 18))))) 6 (1 90) 17 13))
;(setf lst2 '(2 6 90 17 10 22 7 33))
;
;(print (solution lst lst2))


;N3 (a)
;Все числовые элементы исходного смешанного одноуровнего списка удвоить, если
;если сумма его первых числовых элементов больше 10, иначе уменьшить на 10

;Тут желательно переделать, чтобы останавливалась функция раньше времени.
(defun first_n_sum(lst n)
   (cond ((or (null lst) (< n 1)) 0)
         ((and (numberp (car lst)) (> n 0)) (+ (car lst) (first_n_sum (cdr lst) (- n 1))))
         (t (first_n_sum (cdr lst) n)) ))

(setq a '(sd 35 fh iu d w 43 veg 67 fgh 2 4 5))
```

```
;(print (first_n_sum a 3))

(defun double_nums(lst)
   (cond ((null lst) nil)
        ((numberp (car lst)) (cons (* 2 (car lst)) (double_nums (cdr lst))) )
        (t (cons (car lst) (double_nums (cdr lst)))) ))

(defun sub_nums(lst num)
   (cond ((null lst) nil)
        ((numberp (car lst)) (cons (- (car lst) 10) (sub_nums (cdr lst) num)) )
        (t (cons (car lst) (sub_nums (cdr lst) num))) ))



(defun change_nums(lst)
   (cond ((> (first_n_sum lst 2) 10) (double_nums lst))
        (t (sub_nums lst 10)) ))

; Функционалы
(defun sum-check (lst)
   (reduce (lambda(a x) (if (eq a 4) break) (list (car nlst) (cadr nlst)))

   )

(defun calc (lst)
       (let ((sc (sum-check lst)))
            (mapcar #'(lambda (x) (if (numberp x) (if (> sc 10) (* x 2) (- x 10)) x)) lst)))

(setf a '(d a t 4 yt 5 12 w))
;
(print (calc a))

;N3 (b)
;Даны два структурированных смешанных списка. Получить из этих списков
;числовые множества (одноуровневые списки) и найти пересечение этих двух
;множеств.

(defun intersect_mul_lvl(lst1 lst2)
   (intersect_func (sort_ins (into_one lst1 nil)) (sort_ins (into_one lst2 nil))) )

; Функционалы
( defun intersect_func (lst1 lst2)
   (mapcan (lambda (x) (cond ((member x lst2 :test #'equalp) (list x)))) lst1)
```

```
   )
(setf lst '(3 2 5 ( 4 ( ( 7 )(((9 11 18))))) 6 (1 90) 17 13))
(setf lst2 '(2 (6 90) (((17) ((10 22)) 7)) 33))

;print (intersect_mul_lvl lst lst2))

;N4 (a)
;Реализовать выделение из ассоциативной таблицы с числовыми ключами элементов,
;стоящих на нечетных позициях и уменьшить все ключи результирующей таблицы
;на количество элементов в ней

;Выделить из таблицы... Ты ебанутая?

(defun from_odd_pos(lst idx)
   (cond ((null lst) nil)
         ((eq 0 idx) (from_odd_pos (cdr lst) 1))
         (t (cons (car lst) (from_odd_pos (cdr lst) 0))) ))

(defun sub_keys(lst num)
   (cond ((null lst) nil)
         (t (cons  (cons (- (caar lst) num) (cdar lst)) (sub_keys (cdr lst) num))) ))

(defun pick_out(lst)
   (let ( (out (from_odd_pos lst 1)) )
      (sub_keys out (length out)) ))

(setf a '( (3 . 5 ) (6 . 7) (1 . 5) (8 . 2) (2 . 90)))

; Функционалы
(setf table '((1 . 4) (4 . 6) (7 . 8) (9 . 4)))

(defun len (table)
         (reduce #'(lambda (x y) (1+ x)) table :initial-value 0)
)
(defun change (table)
         (
                  mapcar #'(lambda (x) (cons (- (car x) (len table)) (cdr x))) table
         )
)
```

```
(defun from_odd_pos_func (table)
   (setf pos 0)
                (
                   mapcan (lambda (x) (setf pos (+ 1 pos)) (cond ((oddp pos)  (list x))
                            ) ) table
           )
)


;(print (pick_out a))
;(print (from_odd_pos_func a ))

;N4 (b)
;Дан смешанный структурированный список. Реализовать выделение из списка
;числовых атомов, лежащих в заданном интервале, и символьных атомов в
разные списки

;Что-то не совсем понятно о каких интервалах идет речь

(defun lists_from_lst(lst int_num int_symb)
   (let ((lst_one (into_one lst nil)))
      (cons (get_nums lst_one 0 (- (car int_num) 1) (cdr int_num)) (get_symbols
lst_one 0 (- (car int_symb) 1) (cdr int_symb))) ))

(defun get_nums(lst curr s e)
   (cond ((null lst) nil)
         ((not (numberp (car lst))) (get_nums (cdr lst) curr s e))
         ((or (< curr s) (>= curr e)) (get_nums (cdr lst) (+ 1 curr) s e))
         (t (cons (car lst) (get_nums (cdr lst) (+ 1 curr) s e))) ))

(defun get_symbols(lst curr s e)
   (cond ((null lst) nil)
         ((not (symbolp (car lst))) (get_symbols (cdr lst) curr s e))
         ((or (< curr s) (>= curr e)) (get_symbols (cdr lst) (+ 1 curr) s e))
         (t (cons (car lst) (get_symbols (cdr lst) (+ 1 curr) s e))) ))


;(setf lst '(3 2 (b fd 5) 5 ( 4 ( ( 7 )(((9 11 bfd a 18 l))))) 6 (1 90) pok 17 lop 13))
;
;(print (lists_from_lst lst '(4 . 7) '(3 . 6)))
```

```
;Удаление дубликатов в списке. Сортируем и делаем следующее:
(defun remove_dup(lst prev)
   (cond ((null lst) nil)
         ((eq (car lst) prev) (remove_dup (cdr lst) prev))
         (t (cons (car lst) (remove_dup (cdr lst) (car lst)))) ))

;(setf a '(43 50 51 52 55 55 100))
;(print (remove_dup a "inf"))


;Вычисление n!! (Если n - четное, то n!! = 2 * 4 * ... * n, если n - нечетное, то n!! =
1 * 3 * ... * n)
(defun factorial_ii_help(n res)
   (cond ((< n 2) res)
         (t (factorial_ii_help (- n 2) (* n res))) ))

(defun factorial_ii(n)
   (factorial_ii_help n 1))

;(print (factorial_ii 10))

;(print (member-if #'listp '(a b nil nil c d)))
;
;(print (reduce (lambda(a b c)(+ a (/ b 50) (/ c 50))) '(1 5 3) '(1 5 3) :initial-value 0 ))
;
;(print (reduce (lambda (a x) (if (evenp x) (+ a x) a)) '(1 1 1 2 4 6 7 9 11) :initial-value
0))


;(print (reduce #'+ '(4 5 7 89 3)))
```

---

1. B1. В структурированном (сложном списке, в котором другие списки) один заданный элемент заменить на другой. Например 1 на 6.

```
(setf lst1 (2 (10 2 3) 4 5 1 () (2 3 2) 2))

(defun func_cov (lst a b) (func lst a b lst))

(defun func (lst a b lst_help)
        (
                cond ((null lst) lst_help)
```

```
                ((listp (car lst)) (and (func (car lst) a b lst_help) (func (cdr lst) a b lst_help)))
                ((= (car lst) a) (func (cdr (rplaca lst b)) a b lst_help))
                (t (func (cdr lst) a b lst_help))
        )
)

(defun ffunc (lst a b)
        (mapcar #'(lambda (x) (if (listp x) (ffunc x a b) (if (= x a) b x))) lst)
)
```

2. посчитать среднее значение чисел в списке, которые на нечетных позициях. Когда используешь функционалы, рекомендуют делать с редусом и скакойто его настройкой инит-валю-чего-то-там, чтоб было эффективно.

```
(setf lst1 '(2 (10 2 3) 4 5 1 () (2 3 2) 2))
;(setf lst1 '(() (((5))) 1 1 1))
(setf one-lvl '( 1 (2 (3 (4)))))

;( 1 ( 2 ( 3 ( 4 ))))

(setf lst2 '((1 9) (9) 2))

(defun avg_res (lst) (/ (car lst) (cadr lst)))
(defun avg_cov (lst) (avg_res (avg lst 1 0 0)))
(defun avg(lst pos count sum)
        (
                cond ((null lst) (list sum count))
        ((and (listp (car lst)) (EVENP pos))( list
                                (+ sum (car (avg (cdr lst) (+ pos 1) 0 0)))
                                (+ count (cadr (avg (cdr lst) (+ pos 1) 0 0)))
                               ))
        ((listp (car lst)) (list (+ sum
                  (car (avg (cdr lst) (+ pos 1) 0 0))
                  (car (avg (car lst) 1 0 0))) (+ count
                  (cadr (avg (cdr lst) (+ pos 1) 0 0))
                  (cadr (avg (car lst) 1 0 0)))
                  )
                )
                        ((ODDP pos) (avg (cdr lst) (+ pos 1) (+ count 1) (+ sum (car lst))))
                        (t (avg (cdr lst) (+ pos 1) count sum))
        )
)



(defun avg_cov (lst) (avg lst 0 0 0))
(defun avg(lst pos count sum)
        (
                cond ((null lst) (/ sum count))
                        ((oddp pos) (avg (cdr lst) (+ pos 1) (+ count 1) (+ sum (car lst))))
                        (t (avg (cdr lst) (+ pos 1) count sum))
        )
)
```

Билет 2

Найти A/B где A и B числовые множества полученные из 2х одноуровневых списков(хз зачем это)

```
(defun f (lst1 lst2)
   (remove-if (lambda (x) (member x lst2 :test #'equalp )) lst1)
)

(defun f(lst1 lst2 res)
   (let ((a (car lst1)))
         (if(cdr lst1)
               (if
                     (member a lst2 :test #'equalp)
                     (f (cdr lst1) lst2 res)
                     (f (cdr lst1) lst2 (cons a res))
               )
               res
         )

   )
)
```

2. Удаление элементов в структурированном списке.

```
(defun myrem (lst el)
   (let((ls (delete el lst :test #' equalp)))
         (cond
               ((null ls) nil)
               ((listp(car ls))(cons(myrem (car ls) el)(myrem (cdr ls) el)))
               (T(cons (car ls)(myrem (cdr ls) el)))
         )
   )
)

(defun myfunk(X El)
   (if (equalp x el)
         nil
         (if(listp x)
               (list (myrem x el))
               (list x)
         )
   )
)

(defun myrem (lst el)
   (mapcan #'(lambda(X)(myfunk X El)) lst
   )
)
```

Билет 8

1. **Реализовать добавление в ассоциативный список нескольких точечных пар, заданных списком ключей и списком значений. (своя функция)**

   С рекурсией:

```
(setq lst_help '((1 . 1val)(2 . 2val)(3 . 3val)(4 . 4val)))

(defun add_to_lst (lst couple)
	(nconc lst (list couple))
	(print lst)
)

(defun add_item (lst a b)
	(add_to_lst lst (cons a b))
)

(defun add_items (lst key_list val_list)
	(if (car key_list)
		(and (add_item lst (car key_list) (car val_list))
		    (add_items lst (cdr key_list) (cdr val_list))
		)
		(print lst)
	)

)
```

С функционалом:

```
(defun add_items2 (lst key_list val_list)
	(nconc lst (mapcar #'(lambda (x y) (cons x y)) key_list val_list))
)
```

2. Дан смешанный структурированный список. Получить отсортированный по возрастанию список из числовых элементов исходного списка, входящих в заданное в виде одноуровнего смешанного списка множество.

```
(setq task_help '((3 2 1) f h (t 6) ((u) 7 0)))
(defun into_one (lst rst)
	(cond ((null lst) rst)
		((atom lst) (cons lst rst))
		(t (into_one (car lst) (into_one (cdr lst) rst)) )
	)
)

(defun into_one_level (lst)
	(into_one lst ())
)

(defun get_num_list (lst)
	(remove-if-not #'numberp (into_one_level lst))
)

(defun insert_help (x lst)
	(cond ((null lst) (list x))
		((<= x (car lst)) (cons x lst))
		(t (cons (car lst) (insert_help x (cdr lst))))
	)
```

```
)
(defun sort_help (lst1 lst2)
	(cond ((null lst1) lst2)
		(t (sort_help (cdr lst1) (insert_help (car lst1) lst2)))
	)
)

(defun my_sort (lst)
	(sort_help lst ())
)

(defun task2(lst)
	(my_sort (get_num_list lst))
)
```

Примечание: тут есть и рекурсии, и функционалы. Признаться честно, я не в курсе, как разворачивать вложенные списки функционалом в обход рекурсии - в голове все равно получается мысль о рекурсии функционалов. Так что здесь имеет место сделать ссылку на то, что в задании не сказано двумя способами делать. Если будут идеи, готов помочь реализовать.

Билет 1

1. **Все числовые элементы исходного смешанного одноуровневого списка удвоить, если сумма его первых двух числовых элементов больше 10 и уменьшить на 10 в противном случае (Использовать функционалы, Использовать рекурсию)**

Рекурсия:

```
(defun check-help (lst d)
    (cond ((endp lst) 0)
        ((and (numberp (car lst)) (eql d 1)) (car lst))
        ((and (numberp (car lst)) (eql d 0)) (+ (car lst) (check-help (cdr lst) 1)))
        (t (check-help (cdr lst) d))))

(defun sum-check (lst)
    (check-help lst 0))

(defun func (cnd x)
    (if (> cnd 10)
        (* x 2)
        (- x 10)))

(defun calc-help (lst sc)
    (cond ((null lst) nil)
        ((numberp (car lst)) (cons (func sc (car lst)) (calc-help (cdr lst) sc)))
        (t (cons (car lst) (calc-help (cdr lst) sc)))))

(defun calc (lst)
    (calc-help lst (sum-check lst)))
```

Функционалы:

```
(defun callme (lst)
( sum-check (remove-if-not #'numberp lst))
)

(defun sum-check (nlst)
        (reduce #'+ (list (car nlst) (cadr nlst)))
         )
)

(defun calc (lst)
    (let ((sc (callme lst)))
        (mapcar #'(lambda (x) (if (numberp x) (if (> sc 10) (* x 2) (- x 10)) x)) lst)))
```

Билет 9

1. **Все числовые элементы исходного смешанного одноуровневого списка удвоить, если сумма его первых двух числовых элементов больше 10 и уменьшить на 10 в противном случае (Использовать функционалы, Использовать рекурсию)**

```
        (defun all_minus_10 (lst)
        (mapcar #'(lambda (x)
                                    (cond ((numberp x) (- x 10))
                                            ((listp x) (all_minus_10 x))
                                            (T x)
                                    )
        )
        lst
        )
        )

(defun all_mult_2 (lst)
        (mapcar #'(lambda (x)
        (cond ((numberp x) (* x 2))
                ((listp x) (all_mult_2 x))
                (T x)
        )
        )
        lst
        )
)

(defun two_number_10 (lst sum)
        (cond
                (( and (numberp (car lst)) (> sum 0) ) (+ sum (car lst)))
                ((numberp (car lst)) (two_number_10 (cdr lst) (+ sum (car lst))) )
                (T (two_number_10 (cdr lst) sum))
        )
)

(defun task1 (lst)
```

```
	(cond ( (equal lst NIL) NIL )
		( (> (two_number_10 lst 0) 10) (all_mult_2 lst) )
		( (<= (two_number_10 lst 0) 10) (all_minus_10 lst) )
	)
)
```

2. Даны два структурированных смешанных списка. Получить из этих списков числовые множества (одноуровневые списки) и найти пересечение этих двух множеств (Использовать функционалы, Использовать рекурсию.)

```
(defun in-list (a lst)
	(cond
		((null lst) NIL)
		((eq a (car lst)) T)
		(T (in-list a (cdr lst)))
	)
)

(defun mult_of_lists (a b)
	(cond
		((null a) NIL)
		((null b) NIL)
		((in-list (car a) b) (cons (car a) (mult_of_lists (cdr a) b)) )
		(T (mult_of_lists (cdr a) b))
	)
)

(defun lst_to_enum (lst enum)
	(mapcan #'(lambda (x)
		(cond ((numberp x) (cons x enum))
			(T NIL)
		)
	)
	lst
	)
)

(defun task2 (lst1 lst2)
(mult_of_lists (lst_to_enum lst1 NIL) (lst_to_enum lst2 NIL))
)
```

Билет 11.

Вычислить n!!, где n - кол-во чисел в одноуровневом смешанном списке

Рекурсия:

```
(defun d_f_r (n)
	(if
		(> n 3)
		(* n (d_f_r (- n 2)))
		n
	)
)
```

Сделал через reduce, предварительно создав список. Увы, но больше идей в голову не приходит, как реализовать эту программу через функционалы. Проще через рекурсию делать)

```
(defun my_cons (n)
	(if
		(> n 1) (cons n (my_cons (- n 2)))
	)
)

(defun d_f_f (n)
	(if
		(> n 3) (reduce #'* (my_cons n))
		n
	)
)
```

Билет 15.

Выделить из одноуровнего, смешаного списка числа, сформировать из них множество (в виде одноуровнего списка) и найти количество элементов в нем.

Рекурсия:

```
(defun len (lst)
 (cond ((null lst) 0)
  (t (+ 1 (len (cdr lst))))
 )
)

(defun my_union (lst count)
	(cond
		((null lst) (AND (print count) Nil))
		((numberp (car lst))
			(union (cons (car lst) nil) (my_union (cdr lst) )))
		(t (my_union (cdr lst) count))
	)
)
```

Функционал:

```
(defun len (lst)
	(reduce #'(lambda (x y) (1+ x)) lst :initial-value 0)
)

(defun my_union_func (lst)
	(remove-if #'null

	(maplist #'(lambda (x)
				(cond
					((member (car x) (cdr x)) Nil)
					(t (car x))
				))

	(remove-if #'(lambda (x)
						(cond
							((symbolp x) t)
							(t Nil)
						)
				)
	lst)
 )))
```

Без билетные:

Есть два смешанных множества, оставить в обоих только числа и вычислить их объединение

Функционал:

```
(defun operset(A B)
	(union	(remove-if #'(lambda(x)(null x)) (mapcar #'(lambda(x)(if(numberp x)x)) A))
			(remove-if #'(lambda(x)(null x)) (mapcar #'(lambda(x)(if(numberp x)x)) B))
	)
)
```

Рекурсия:

```
(defun opersetR(A B)
	(union
		(if (not(null A))
			(if (numberp (car A))
				(cons (car A) (operset (cdr A) B))
				(operset (cdr A) B)
			)
			()
		)
		(if (not(null B))
			(if (numberp (car B))
				(cons (car B) (operset A (cdr B)))
				(operset A (cdr B))
			)
			()
		)
	)
)
```

Билет 2

Найти A/B где A и B числовые множества полученные из 2х одноуровневых списков(хз зачем это)

```
(defun f (lst1 lst2)
  (remove-if (lambda (x) (member x lst2 :test #'equalp )) lst1)
)

(defun f(lst1 lst2 res)
  (let ((a (car lst1)))
	(if(cdr lst1)
		(if
			(member a lst2 :test #'equalp)
			(f (cdr lst1) lst2 res)
			(f (cdr lst1) lst2 (cons a res))
		)
		res
	)

  )
)
```

Билет 14.
Номер 1. Реализовать выделение из ассоциативной таблицы с числовыми ключами элементов , стоящих на нечетных позиция и уменьшить все ключи результирующей таблицы на количество элементов в ней.
Рекурсив:

```
(defun get-odd (lst)
	(cond
		((null lst) Nil)
		((null (cdr lst)) Nil)
		(t (cons (car lst) (get-odd (cddr lst))))
	)
)

(defun mi10 (lst head vichitaemoe)
	(cond
		((null lst) head)
		(t (and (rplaca lst `(,(- (caar lst) vichitaemoe) . ,(cdar lst))) (mi10 (cdr lst)
head vichitaemoe)))
	)
)

(defun len (table)
 (cond ((null table) 0)
  (t (+ 1 (len (cdr table))))
 )
)

(defun bil6zad1rec (table)
	(let
		(
			(mita (get-odd table))
		)
		(mi10 mita mita (len mita))
	)
)
```

функционалы:

```
(setf table '((1 . 4) (4 . 6) (7 . 8) (9 . 4)))

(defun len (table)
```

```
	(reduce #'(lambda (x y) (1+ x)) table :initial-value 0)
)

(defun change (table)
	(
		mapcar #'(lambda (x) (cons (- (car x) (len table)) (cdr x))) table
	)
)
```

Билет 15.

Выделить из одноуровнего, смешаного списка числа, сформировать из них множество (в виде одноуровнего списка) и найти количество элементов в нем.

Рекурсия:

```
(defun len (lst)
 (cond ((null lst) 0)
  (t (+ 1 (len (cdr lst))))
 )
)

(defun my_union (lst count)
	(cond
		((null lst) (AND (print count) Nil))
		((numberp (car lst))
			(union (cons (car lst) nil) (my_union (cdr lst) )))
		(t (my_union (cdr lst) count))
	)
)
```

Функционал:

```
(defun len (lst)
	(reduce #'(lambda (x y) (1+ x)) lst :initial-value 0)
)

(defun my_union_func (lst)
	(remove-if #'null

	(maplist #'(lambda (x)
				(cond
					((member (car x) (cdr x)) Nil)
					(t (car x))
				))

	(remove-if #'(lambda (x)
					(cond
						((symbolp x) t)
						(t Nil)
					)
				)
	lst)
 )))
```