НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«Московский энергетический институт»
Кафедра математического и компьютерного моделирования

«Структуры данных и методы программирования»
Отчет по курсовой работе
«Программирование на языках Lisp,
FRL и Prolog»

Вариант №3

Выполнил: Волков П.Е.
Группа: А-14-19

Преподаватель: Чернов П.Л.

Москва 2021г

Часть 1. Язык LISP

1. Реализовать функции (@FINDLIST object list test) и (@FINDLIST-IF test list) , осуществляющие поиск на верхнем уровне в списке list. В качестве результата возвращается список элементов из list, для которых (test object element) не NIL. Здесь element - текущий элемент списка list. Если аргумент test опущен, то test = EQUAL. Для второй функции предикат test является одноместным.
Примеры: (@FINDLIST 'A '(B C A B C A)) => (A A)
(@FINDLIST 5 '(1 0 -5 6 10) '<) => (6 10)
(@FINDLIST-IF 'MINUSP '(5 10 -3 -4)) => (-3 -4)

Код программы:

```
(defun _run_test (obj lst test res) (
   (cond ((null (car lst)) res)
         ((not (null (funcall test obj (car lst)))) (_run_test obj (cdr lst) test (cons (car lst) res)))
         (T (_run_test obj (cdr lst) test  res)))))

(defun @findlist (obj lst test) (
   (cond ((or (null lst) (not (listp lst))) nil)
         ((null test) (_run_test obj lst 'eq '()))
         (T (_run_test obj lst test '())))))

(defun _findlist-if (test lst res) (
   (cond ((null lst) res)
         ((funcall test (car lst)) (_findlist-if test (cdr lst) (cons (car lst) res)))
         (T (_findlist-if test (cdr lst) res)))))

(defun @findlist-if (test lst) (
   (cond ((or (null lst) (not (listp lst))) nil)
         (T (_findlist-if test lst '())))))
```

---

2. Реализовать функцию (@INSEND atom list), возвращающую список list, в котором в конец каждого подсписка добавлен атом atom.
Пример: (@INSEND 'A '((B C) (B (C) D) (C D) NIL))
((B C A) (B (C A) D A) (C D A) (A))

(Для удобства и избежания ненужного копирования внешней функции был добавлен 3-й аргумент к функции @insend, которым является одна из написанных функций:
_ins1 - рекурсивная
_ins2 - итерационная
_ins3 - с использованием функционалов

Код программы:

```
(defun @insend (atm lst func) (
   (cond ((null lst) nil)
         ((and (atom atm) (listp lst)) (funcall func atm lst '()))
         (T nil))))

(defun _ins1 (atm lst) (
   (cond ((null lst) (list atm))
         ((atom (car lst)) ((cons (car lst) (_ins1 atm (cdr lst)))))
         (T (append (list (_ins1 atm (car lst))) (_ins1 atm (cdr lst)))))))


(defun _ins2 (atm lst res) (
   (loop ((null (car lst)) (setq res (append res (list atm))))
         (setq cur_elem (pop lst))
         (cond ((atom cur_elem) (setq res (append res (list cur_elem))))
               (T (setq res (append res (list (_ins2 atm cur_elem '()))))))))

(defun _ins3 (atm lst) (
   (defun __ins0 (lst) (
      (cond ((null (cdr lst)) atm)
            ((listp (car lst)) (_ins3 atm (car lst)))
            (T (car lst)))))
   (maplist '__ins0 (append lst (list nil)))))
```

##

Часть 2. Язык FRL

3. Реализовать функцию (ANALYSIS sentence), позволяющую распознать синтаксически правильные предложения русского языка. Под синтаксически правильными предложениями будем понимать те, в которых правильный порядок слов и все слова предложения имеются в словаре.

Код программы:

```
(deframeq vocabulary
   (nouns ($value (avtomobil) (stol) (victor) (on)))
   (adjectives ($value (bystriy) (kholodniy) (priyatniy) (zeleniy)))
   (verbs ($value (bejit) (gorit) (molchit) (smotrit)))
)

(defun is_noun (word)
```

```
    (cond ((member word (fget vocabulary nouns $value)) T)
        (T nil)))

(defun is_adjective (word)
    (cond ((member word (fget vocabulary adjectives $value)) T)
        (T nil)))

(defun is_verb (word)
    (cond ((member word (fget vocabulary verbs $value)) T)
        (T nil)))

(defun is_word (word)
    (cond ((or (is_noun word) (is_adjective word) (is_verb word)) T)
        (T nil)))

(defun analysis (sent prev_word) (
    (cond ((null sent) (is_verb prev_word))
        ((not (is_word (car sent))) nil)
        (T (cond ((or (and (is_adjective prev_word) (is_verb (car sent)))
                    (and (is_noun prev_word) (not (is_verb (car sent)))))
                  nil)
                (T (analysis (cdr sent) (car sent))))))
))
```

##

Часть 3. Язык PROLOG

Код программы:

```
domains
    list = integer*
    stringlist = string*
predicates
    pred(integer)
    findlist(integer, list, list)
    findlist_if(list, list)

    noun(string)
    adj(string)
    verb(string)
    rule(string, string)
    analysis(stringlist)

clauses
```

```
noun("avtomobil").
noun("stol").
noun("victor").
noun("on").

adj("bystriy").
adj("kholodniy").
adj("priyatniy").
adj("zeleniy").

verb("bejit").
verb("gorit").
verb("molchit").
verb("smotrit").

pred(X) :- X < 0.

findlist(_, [], []).
findlist(Obj, [Head | Tail], [Head | Res]) :- Obj < Head, findlist(Obj, Tail, Res).
findlist(Obj, [Head | Tail], Res) :- findlist(Obj, Tail, Res), !.

findlist_if([], []).
findlist_if([Head | Tail], [Head | Res]) :- pred(Head), findlist_if(Tail, Res).
findlist_if([Head | Tail], Res) :- findlist_if(Tail, Res), !.

rule(W1, W2) :- adj(W1), adj(W2).
rule(W1, W2) :- adj(W1), noun(W2).
rule(W1, W2) :- noun(W1), verb(W2).

analysis([]).
analysis([Word]) :- verb(Word).
analysis([W1, W2 | Sent]) :- rule(W1, W2), analysis([W2 | Sent]).
```