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» (ИУ7)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04 Программная инженерия**

О Т Ч Е Т

по лабораторной работе № 2 0

Дисциплина: Функциональное и логическое программирование

Студент ИУ7-62Б _____ Е.В. Брянская
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ Н.Б.Толпинская
Ю.В.Строганов
(Подпись, дата) (И.О. Фамилия)

Москва, 2021

Задание

Используя хвостовую рекурсию, разработать эффективную программу (комментируя назначение аргументов), позволяющую:

1. Сформировать список из элементов числового списка, больших заданного значения
2. Сформировать список из элементов, стоящих на нечетных позициях исходного списка (нумерация от 0)
3. Удалить заданный элемент из списка (один или все вхождения)
4. Преобразовать список в множество (можно использовать ранее разработанные процедуры)

Убедиться в правильности результатов.

Для одного из вариантов вопроса и 1-ого задания составить таблицу, отражающую конкретный порядок работы системы.

Так как резольвента хранится в виде стека, то состояние резольвенты следует отображать в столбик: вершина – сверху! Новый шаг надо начинать с нового состояния резольвенты! Для каждого запуска алгоритма унификации, требуется указать № выбранного правила и дальнейшие действия – и почему.

```
domains
        lst = integer*.
predicates
        create_lst_morethan(lst, integer, lst).
        create_lst_morethan(lst, integer, lst, lst).
        create_lst_morethan_2(lst, integer, lst).

        create_lst_oddpos(lst, lst).
        create_lst_oddpos(lst, lst, lst).
        create_lst_oddpos_2(lst, lst).

        delete_element_all(lst, integer, lst).
        delete_element_all(lst, integer, lst, lst).
        delete_element_all_2(lst, integer, lst).

        create_set(lst, lst).
        create_set(lst, lst, lst).
        create_set_2(lst, lst).
clauses
        % create list from elements that more than number
        % 2. optimized
        create_lst_morethan_2([X|T], Num, [X|Res_tail]) :-      % X - 1st element [index = 0]
                X > Num,                                        % T - tail
                create_lst_morethan_2(T, Num, Res_tail),        % Num - control number
                !.                                              % Res_tail - result tail
        create_lst_morethan_2([_|T], Num, Res_tail) :-
                create_lst_morethan(T, Num, Res_tail),
                !.
        create_lst_morethan_2([], _, []) :- !.

        % 1. non-optimized
        create_lst_morethan([X|T], Num, Res_temp, Res) :-       % X - 1st element [index = 0]
                X > Num,                                        % T - tail
                Temp = [X|Res_temp],                            % Num - control number
                create_lst_morethan(T, Num, Temp, Res),         % Res_temp - temporary list
                !.                                              % Res - result
        create_lst_morethan([_|T], Num, Res_temp, Res) :-       % Temp - temporary var
                create_lst_morethan(T, Num, Res_temp, Res),     % Lst - list
                !.
        create_lst_morethan([], _, Res, Res) :- !.
```

```
create_lst_morethan(Lst, Num, Res) :-
        create_lst_morethan(Lst, Num, [], Res),
        !.

%-------------------------------------------------------------
% create list from elements that stands in the odd positions
% 1. non-optimized
create_lst_oddpos([_, X|T], Res_temp, Res) :-          % X - 2nd element [index = 1], T - tail
        Temp = [X|Res_temp],                           % Res_temp - temporary list, Res - result
        create_lst_oddpos(T, Temp, Res),               % Temp - temporary var
        !.
create_lst_oddpos([_], Res, Res) :- !.
create_lst_oddpos([], Res, Res) :- !.
create_lst_oddpos(Lst, Res) :-
        create_lst_oddpos(Lst, [], Res),
        !.

% 2. optimized
create_lst_oddpos_2([_, X|T], [X|Res_tail]) :-         % X - 2nd element [index = 1], T - tail
        create_lst_oddpos_2(T, Res_tail),              % Res_tail - result list
        !.
create_lst_oddpos_2([_], []) :- !.
create_lst_oddpos_2([], []) :- !.

%-------------------------------------------------------------
% delete element (all occurrence)
% 1. non-optimized
delete_element_all([X|T], Num, Res_temp, Res) :-   % X - 1st element [index = 0], T - tail, Num - control number
        X = Num,                                       % Res_temp - temporary list, Res - result
        delete_element_all(T, Num, Res_temp, Res),     % Lst - list
        !.                                             % Temp - temporary var
delete_element_all([X|T], Num, Res_temp, Res) :-
        Temp = [X|Res_temp],
        delete_element_all(T, Num, Temp, Res),
        !.
delete_element_all([], _, Res, Res) :- !.
delete_element_all(Lst, Num, Res) :-
        delete_element_all(Lst, Num, [], Res),
        !.

% 2. optimized
delete_element_all_2([Num|T], Num, Res_tail) :-        % Num - control number
        delete_element_all_2(T, Num, Res_tail),        % Res_tail - result list
        !.                                             % X - 1st element [index = 0]
delete_element_all_2([X|T], Num, [X|Res_tail]) :-      % T - tail
        delete_element_all_2(T, Num, Res_tail),
        !.
delete_element_all_2([], _, []) :- !.

%-------------------------------------------------------------
% create set from list
% 1. non-optimized
create_set([X|T], Res_temp, Res) :-        % X - 1st element [index = 0], T - tail, Res_temp - temporary list
        Temp = [X | Res_temp],             % Res - result
        delete_element_all(T, X, T_cor),   % Temp - temporary var
        create_set(T_cor, Temp, Res),      % T_cor - tail after delelition all elements = X
        !.                                 % Lst - list
create_set([], Res, Res) :- !.
create_set(Lst, Res) :- create_set(Lst, [], Res).

% 2. optimized
create_set_2([X|T], [X|Res_tail]) :-           % X - 1st element [index = 0], T - tail
        delete_element_all_2(T, X, Temp),      % Res_tail - result list
        create_set_2(Temp, Res_tail),          % Temp - tail after delelition all elements = X
        !.
```

```
        create_set_2([], []) :- !.
goal
        %create_lst_morethan([10, 0, -5, 0, 3], 0, Result).
        %create_lst_morethan_2([10, 0, -5, 0, 3], 0, Result).
        %create_lst_oddpos([10, 0, -5, 1, 9, -9], Result).
        %create_lst_oddpos_2([10, 0, -5, 1, 9, -9], Result).
        %delete_element_all([10, 5, -3, 0, 5], 5, Result).
        %delete_element_all_2([10, 5, -3, 0, 5], 5, Result).
        %create_set([1, 1, 1, 5], Result).
        %create_set_2([1, 1, 1, 5], Result).
```

Текст процедуры:

```
create_lst_morethan_2([X|T], Num, [X|Res_tail]) :-      % X - 1st element [index = 0]
        X > Num,                                        % T - tail
        create_lst_morethan_2(T, Num, Res_tail),        % Num - control number
        !.                                              % Res_tail - result tail
create_lst_morethan_2([_|T], Num, Res_tail) :-
        create_lst_morethan(T, Num, Res_tail),
        !.
create_lst_morethan_2([], _, []) :- !.
```

Вопрос: create_lst_morethan_2([10, 0, -5], 0, Result).

№	Текущая резольвента - ТР	ТЦ, выбираемые правила: сравниваемые термы, подстановка	Дальнейшие действия
0.	create_lst_morethan_2([10, 0, -5], 0, Result)		
1.	10 > 0, create_lst_morethan_2([0, -5], 0, Res_tail_1), !	create_lst_morethan_2([10, 0, -5], 0, Result) = create_lst_morethan_2([X_1|T_1], Num_1, [X_1|Res_tail_1]) (1) **Удача** **Подстановка:** {X_1=10, T_1=[0, -5], Num_1=0, Result=[10|Res_tail_1]}	Прямой ход Изменение резольвенты: 1. применение редукции 2. применение подстановки
2.	create_lst_morethan_2([0, -5], 0, Res_tail_1), !	10 > 0 **Удача** (истинное логическое выражение) **Подстановка:** {X_1=10, T_1=[0, -5], Num_1=0, Result=[10|Res_tail_1]}	Прямой ход Изменение резольвенты: 1. применение редукции 2. применение подстановки
3.	0 > 0, create_lst_morethan_2([-5], 0, Res_tail_3), !, !	create_lst_morethan_2([0, -5], 0, Res_tail_1) = create_lst_morethan_2([X_3|T_3], Num_3, [X_3|Res_tail_3]) (1) **Удача** **Подстановка:**	Прямой ход Изменение резольвенты: 1. применение редукции 2. применение подстановки

		{X_1=10, T_1=[0, -5], Num_1=0, Result=[10|Res_tail_1], X_3=0, T_3=[-5], Num_3=0, Res_tail_1=[0|Res_tail_3]}	
4.	0 > 0, create_lst_morethan_2([-5], 0, Res_tail_3), !, !	0 > 0 **Неудача** (ложное логическое выражение)	**Откат** (к предыдущему состоянию резольвенты)
5.	create_lst_morethan_2([0, -5], 0, Res_tail_1), !	**Подстановка:** {X_1=10, T_1=[0, -5], Num_1=0, Result=[10|Res_tail_1]}	
6.	create_lst_morethan_2([-5], 0, Res_tail_6), ! !	create_lst_morethan_2([0, -5], 0, Res_tail_1) = create_lst_morethan_2([_|T_6], Num_6, Res_tail_6) (2) **Удача** **Подстановка:** {X_1=10, T_1=[0, -5], Num_1=0, Result=[10|Res_tail_1], T_6=[-5], Num_6=0, Res_tail_1=Res_tail_6}	Прямой ход Изменение резольвенты: 1. применение редукции 2. применение подстановки
7.	-5 > 0, create_lst_morethan_2([], 0, Res_tail_7), !, !, !	create_lst_morethan_2([-5], 0, Res_tail_6) = create_lst_morethan_2([X_7|T_7], Num_7, [X_7|Res_tail_7]) (1) **Удача** **Подстановка:** {X_1=10, T_1=[0, -5], Num_1=0, Result=[10|Res_tail_1], T_6=[-5], Num_6=0, Res_tail_1=Res_tail_6, X_7=-5, T_7=[], Num_7=0, Res_tail_6=[- 5|Res_tail_7]}	Прямой ход Изменение резольвенты: 1. применение редукции 2. применение подстановки
8.	-5 > 0, create_lst_morethan_2([], 0, Res_tail_7), !, !, !	-5 > 0 **Неудача** (ложное логическое выражение)	**Откат** (к предыдущему состоянию резольвенты)
9.	create_lst_morethan_2([-5], 0, Res_tail_6), !, !	**Подстановка:** {X_1=10, T_1=[0, -5], Num_1=0, Result=[10|Res_tail_1], T_6=[-5], Num_6=0, Res_tail_1=Res_tail_6}	
10.	create_lst_morethan_2([], 0, Res_tail_10), !, !, !	create_lst_morethan_2([-5], 0, Res_tail_6) = create_lst_morethan_2([_|T_10], Num_10, Res_tail_10) (2)	Прямой ход Изменение резольвенты: 1. применение редукции 2. применение подстановки

		Удача **Подстановка:** {X_1=10, T_1=[0, -5], Num_1=0, Result=[10|Res_tail_1], T_6=[-5], Num_6=0, Res_tail_1=Res_tail_6, T_10=[], Num_10=0, Res_tail_6=Res_tail_10}	
11.	create_lst_morethan_2([], 0, Res_tail_10), !, !, !	create_lst_morethan_2([], 0, Res_tail_10) = create_lst_morethan_2([X|T], Num, [X|Res_tail]) (1) **Неудача** (пустой список)	Прямой ход, переход к следующему правилу.
	create_lst_morethan_2([], 0, Res_tail_10), !, !, !	create_lst_morethan_2([], 0, Res_tail_10) = create_lst_morethan_2([_|T], Num, Res_tail) (2) **Неудача** (пустой список)	Прямой ход, переход к следующему правилу.
	!, !, !, !	create_lst_morethan_2([], 0, Res_tail_10) = create_lst_morethan_2([], _, []) **Удача** **Подстановка:** {X_1=10, T_1=[0, -5], Num_1=0, Result=[10|Res_tail_1], T_6=[-5], Num_6=0, Res_tail_1=Res_tail_6, T_10=[], Num_10=0, Res_tail_6=Res_tail_10, Res_tail_10=[]}	Прямой ход Изменение резольвенты: 1. применение редукции 2. применение подстановки
12.	!, !, !	! **Удача** **Подстановка:** {X_1=10, T_1=[0, -5], Num_1=0, Result=[10|Res_tail_1], T_6=[-5], Num_6=0, Res_tail_1=Res_tail_6, T_10=[], Num_10=0, Res_tail_6=Res_tail_10, Res_tail_10=[]}	**Отсечение** (системный предикат отсечения) Прямой ход Изменение резольвенты: 1. применение редукции 2. применение подстановки
13.	!, !	! **Удача** **Подстановка:** {X_1=10, T_1=[0, -5], Num_1=0, Result=[10|Res_tail_1], T_6=[-5], Num_6=0, Res_tail_1=Res_tail_6, T_10=[], Num_10=0, Res_tail_6=Res_tail_10,	**Отсечение** (системный предикат отсечения) Прямой ход Изменение резольвенты: 1. применение редукции 2. применение подстановки

		Res_tail_10=[]}	
14.	!	! **Удача** **Подстановка:** {X_1=10, T_1=[0, -5], Num_1=0, Result=[10|Res_tail_1], T_6=[-5], Num_6=0, Res_tail_1=Res_tail_6, T_10=[], Num_10=0, Res_tail_6=Res_tail_10, Res_tail_10=[]}	**Отсечение** (системный предикат отсечения) Прямой ход Изменение резольвенты: 1. применение редукции 2. применение подстановки
15.	Резольвента **пуста**	! **Удача** **Подстановка:** {X_1=10, T_1=[0, -5], Num_1=0, Result=[10|Res_tail_1], T_6=[-5], Num_6=0, Res_tail_1=Res_tail_6, T_10=[], Num_10=0, Res_tail_6=Res_tail_10, Res_tail_10=[]}	**Отсечение** (системный предикат отсечения) Прямой ход Изменение резольвенты: 1. применение редукции 2. применение подстановки **Вывод**: Result = [10] **Откат** (пустая резольвента)
16.	create_lst_morethan_2([], 0, Res_tail_10), !, !, !	**Подстановка:** {X_1=10, T_1=[0, -5], Num_1=0, Result=[10|Res_tail_1], T_6=[-5], Num_6=0, Res_tail_1=Res_tail_6, T_10=[], Num_10=0, Res_tail_6=Res_tail_10}	**Откат** (отсечение)
17.	create_lst_morethan_2([-5], 0, Res_tail_6), !, !	**Подстановка:** {X_1=10, T_1=[0, -5], Num_1=0, Result=[10|Res_tail_1], T_6=[-5], Num_6=0, Res_tail_1=Res_tail_6}	**Откат** (отсечение)
18.	create_lst_morethan_2([0, -5], 0, Res_tail_1), !	**Подстановка:** {X_1=10, T_1=[0, -5], Num_1=0, Result=[10|Res_tail_1]}	**Откат** (отсечение)
19.	10 > 0, create_lst_morethan_2([0, -5], 0, Res_tail_1), !	**Подстановка:** {X_1=10, T_1=[0, -5], Num_1=0, Result=[10|Res_tail_1]}	**Откат** (унификация с константой)
20.	create_lst_morethan_2([10, 0, -5], 0, Result)	**Подстановка:** {}	Завершение работы

Вывод: эффективность программы достигнута за счёт хвостовой рекурсии, отсечений и расположения знания в БЗ.

Вопросы

1. Как организуется хвостовая рекурсия в Prolog?

При хвостовой рекурсии все действия сделаны до момента выхода из неё, вызов единственен. Выход из рекурсии организуется с помощью отсечения.

2. Какое первое состояние резольвенты?

Начальное состояние резольвенты – вопрос.

3. Каким способом можно разделить список на части, какие, какие требования к частям?

Список можно разделить на голову и хвост, с помощью символа | ([H|T]). Голова (H) должна состоять из не менее, чем одного элемента, а хвост (T) обязательно должен быть одним списком.

4. Как выделить за один шаг первые два подряд идущих элемента списка? Как выделить первый и третий элемент за один шаг?

[E1, E2|_] – первые два подряд идущих элемента списка

[E1, _, E3|_] – первый и третий элемент списка

5. Как формируется новое состояние резольвенты?

Резольвента меняется в два этапа:

1. В текущей резольвенте выбирается одна из целей, для неё выполняется редукция
2. Затем к резольвенте применяется подстановка, полученная, как наибольший общий унификатор цели и заголовка сопоставимого с ней правила.

6. Когда останавливается работа системы? Как это определяется на формальном уровне?

На формальном уровне это определяется тем, что в резольвенте находится исходный вопрос, для которого вся БЗ просмотрена. То есть система завершает работу в случае, когда все возможные ответы рассмотрены.