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 №20

По предмету: «Функциональное и логическое программирование»

Студент: Лаврова А. А.,
Группа: ИУ7-65Б
Преподаватель: Толпинская Н. Б.
Строганов Ю. В.

Москва, 2020 г

Задание:

Используя хвостовую рекурсию, разработать, комментируя аргументы, эффективную программу, 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).

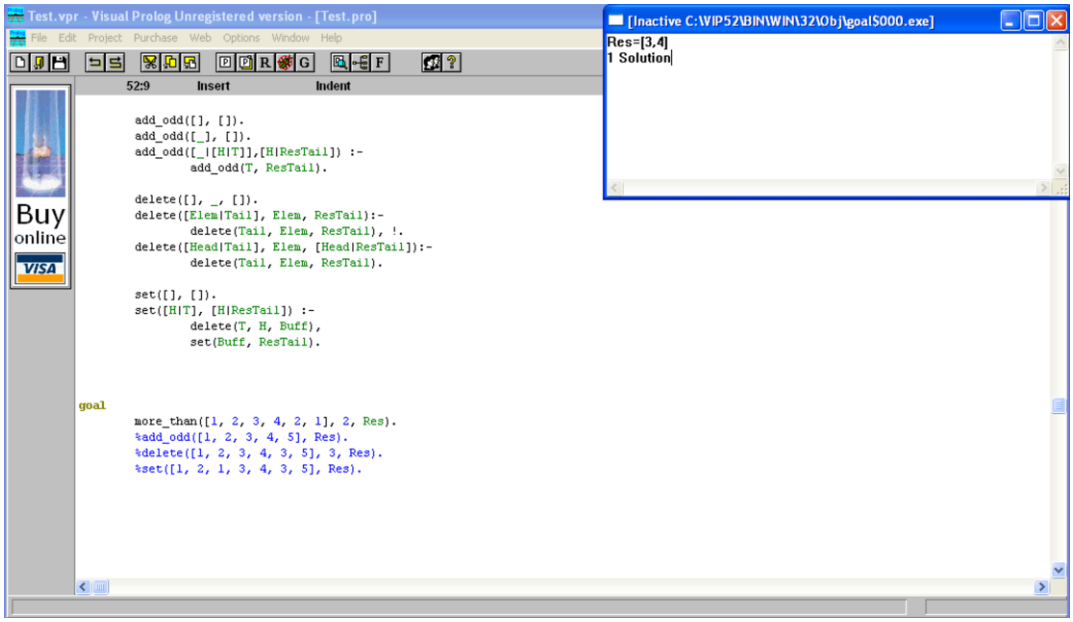
Листинг программы:

```
domains
        list = integer*.

predicates
        more_than(list, integer, list).
        add_odd(list, list).
        delete(list, integer, list).
        set(list, list).

clauses
        more_than([], _, []) :- !.
        more_than([H|T], Num, [H|Tail]) :-
                H > Num,
                more_than(T, Num, Tail), !.
        more_than([H|T], Num, Tail) :-
                H <= Num,
                more_than(T, Num, Tail).

        add_odd([], []) :- !.
        add_odd([_], []) :- !.
        add_odd([_|[H|T]],[H|ResTail]) :-
                add_odd(T, ResTail).

        delete([], _, []):-!.
        delete([Elem|Tail], Elem, ResTail):-
                delete(Tail, Elem, ResTail), !.
        delete([Head|Tail], Elem, [Head|ResTail]):-
                delete(Tail, Elem, ResTail).

        set([], []).
        set([H|T], [H|ResTail]) :-
                delete(T, H, Buff),
                set(Buff, ResTail).



goal
        %more_than([1, 2, 3, 4, 2, 1], 2, Res).
        %add_odd([1, 2, 3, 4, 5], Res).
        %delete([1, 2, 3, 4, 3, 5], 3, Res).
        set([1, 2, 1, 3, 4, 3, 5], Res).
```

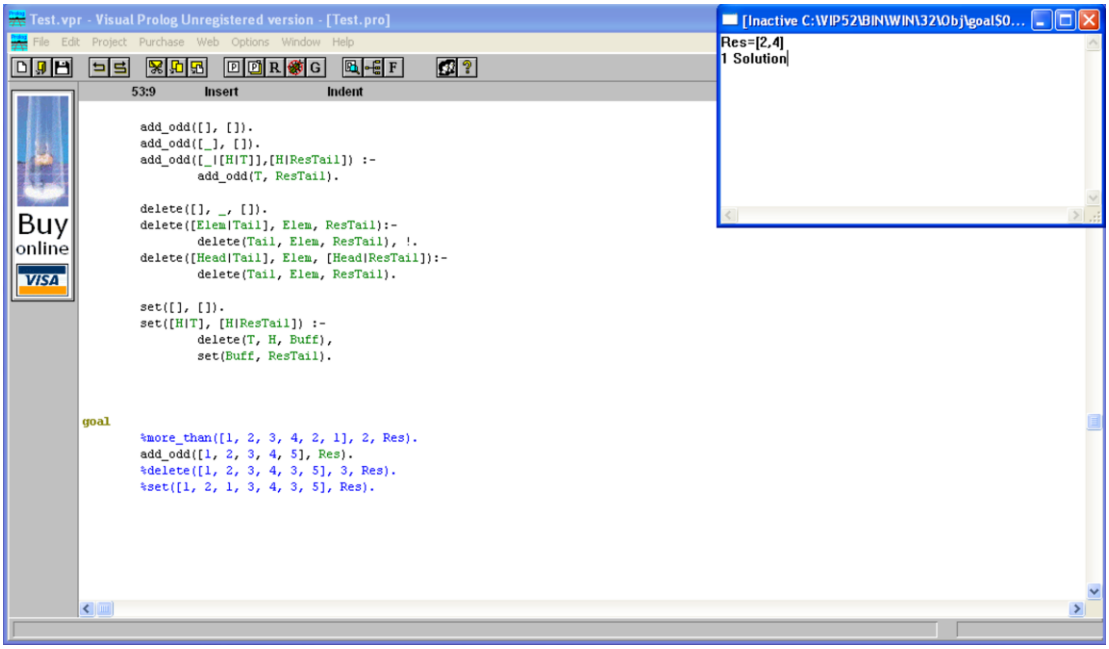
Если все-таки окажется, что отсечение делает рекурсию не хвостовой, то вот немного исправленный вариант:

```
domains
        list = integer*.

predicates
        more_than(list, integer, list).
        add_odd(list, list).
        delete(list, integer, list).
        set(list, list).

clauses
        more_than([], _, []).
        more_than([H|T], Num, [H|Tail]) :-
                H > Num,
                more_than(T, Num, Tail), !.
        more_than([H|T], Num, Tail) :-
                H <= Num,
                more_than(T, Num, Tail).

        add_odd([], []).
        add_odd([_], []).
        add_odd([_|[H|T]],[H|ResTail]) :-
                add_odd(T, ResTail).

        delete([], _, []).
        delete([Elem|Tail], Elem, ResTail):-
                delete(Tail, Elem, ResTail), !.
        delete([Head|Tail], Elem, [Head|ResTail]):-
                delete(Tail, Elem, ResTail).

        set([], []).
        set([H|T], [H|ResTail]) :-
                delete(T, H, Buff),
                set(Buff, ResTail).



goal
        %more_than([1, 2, 3, 4, 2, 1], 2, Res).
        %add_odd([1, 2, 3, 4, 5], Res).
        %delete([1, 2, 3, 4, 3, 5], 3, Res).
        set([1, 2, 1, 3, 4, 3, 5], Res).
```

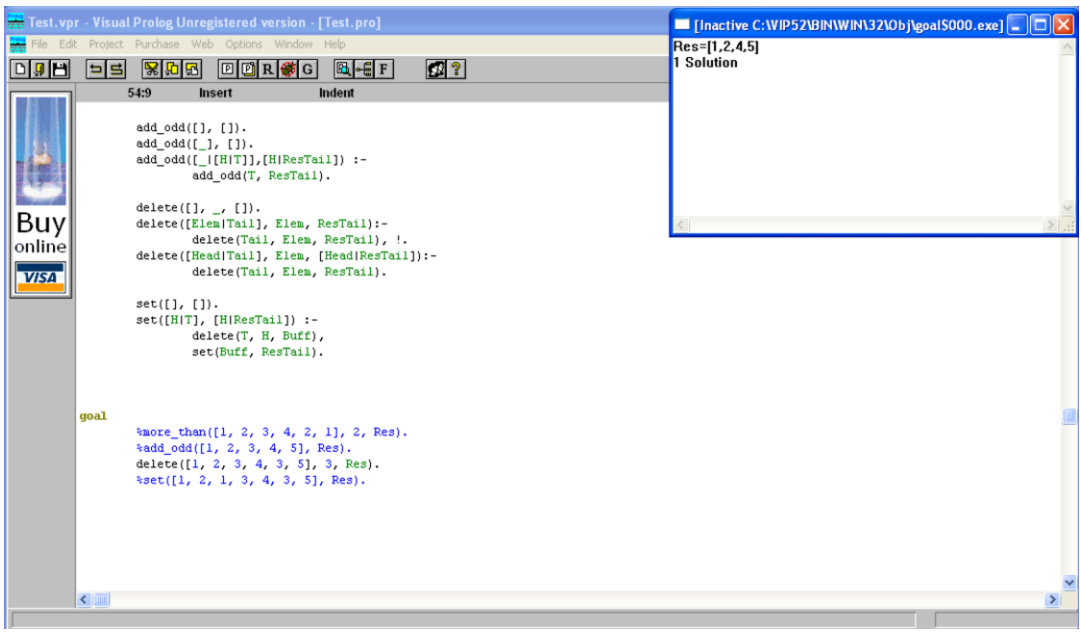
Результаты работы программы:



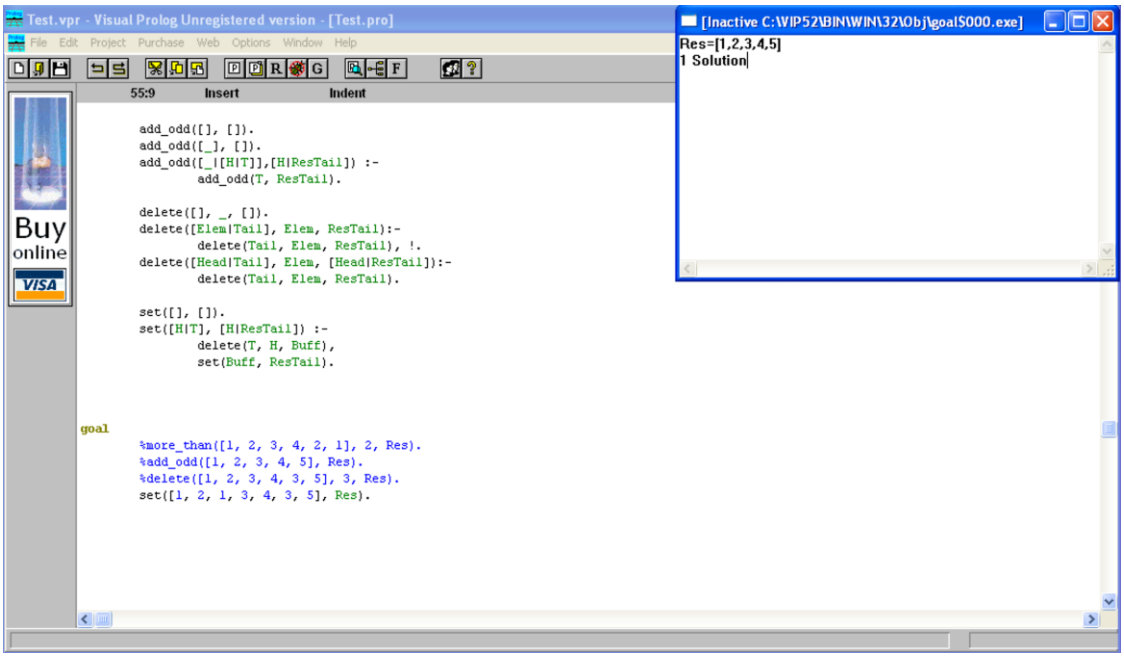
Создание списка из элементов числового списка, больших заданного значения



Создание списка из элементов, стоящих на нечетных позициях исходного списка (нумерация от 0)



Удаление заданного элемента из списка



Преобразование списка в множество

Работа с таблицей:

```
/*1*/more_than([], _, []).
/*2*/more_than([H|T], Num, [H|Tail]) :-
                H > Num,
                more_than(T, Num, Tail), !.
/*3*/more_than([H|T], Num, Tail) :-
                H <= Num,
                more_than(T, Num, Tail).
```

```
more_than([10], 9, Res).
```

№ шага	Текущая резольвента – ТР	ТЦ, выбираемые правила: сравниваемые термы, подстановка	Дальнейшие действия с комментариями
1	more_than([10], 9, Res)	ТЦ: more_than([10], 9, Res)	Поиск знания в БЗ
	more_than([10], 9, Res)	ПР1: [] = [10] _ = 9 [] = Res Неудача	Метка переносится ниже
	more_than([10], 9, Res)	ПР2: [H1|T1] = [10] Min1 = 9 [H1|Tail1] = Res Успех H1 = 10 T1 = [] Min1 = 9 Res = [10|Tail1]	Тело ПР2 заменяет цель в резольвенте
2	more_than([10], 9, Res)	Сравнение: 10 > 9 Успех	
	more_than([10], 9, Res)	ТЦ: more_than([], 9, [10 | Tail1])	Поиск знания с начала БД
3	more_than([], 9, [Tail1])	ПР1: [] = [] Min2 = 9	Пустое тело заменяет цель в резольвенте

		[] = Tail1 Успех []=[] Min2 = 9 Res = [[] | Tail1]	
	-	-	Успех Res=Tail1=[] Возврат к предыдущему состоянию резольвенты
4	more_than([ ], 9, [Tail1])	ПР2: [H2|T2] = [] Min2 = 9 [H2|Tail2] = Res неудача	Метка переносится ниже.
	more_than([ ], 9, [Tail1])	ПР3: [H2|T2] = [] Min2 = 9 [Tail2] = Res неудача	Откат, дошли до конца БЗ, сворачиваем рекурсию
			Res = [10 | Tail1] Tail1 = [] Ответ: [10] Программа завершает работу

Вывод: Эффективность работы программы достигнута за счет использования отсечения, которое ограничивает количество избыточных вычислений. Также использована хвостовая рекурсия, которая помогает оптимизировать использование памяти.

Теоретическая часть

1) *Как организуется хвостовая рекурсия в Prolog?*

   Для осуществления хвостовой рекурсии рекурсивный вызов определяемого предиката должен быть последней подцелью в теле рекурсивного правила и к моменту рекурсивного вызова не должно остаться точек возврата (непроверенных альтернатив). Параметры должны изменяться на каждом шаге так, чтобы в итоге либо сработал базис рекурсии, либо условие выхода из рекурсии, размещенное в самом правиле.

2) *Какое первое состояние резольвенты?*

   Заданный вопрос.

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

   В Prolog для разделения списка на части используется специальный̆ символ «|»:
   [H | T]. Разделение происходит на голову (H) и хвост (T).

4) *Как выделить за один шаг первые два подряд идущих элемента списка? Как выделить 1-й и 3-й элемент за один шаг?*

   Выделить за один шаг два подряд идущих элемента можно с помощью [H1|[H2|_]]. Выделить за один шаг 1-й и 3-й элемент можно с помощью [H1|[_|[H3|_]]].

5) *Как формируется новое состояние резольвенты?*

   Сначала из стека выбирается первая подцель, а затем замена подцели на тело подходящего правила. Потом к полученной конъюнкции применяется подстановка, то есть наибольший общий унификатор.

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

   Система завершает работу в случае, если найдены все возможные ответы на вопрос. На формальном уровне – если в резольвенте находится исходный вопрос, для которого пройдена вся БЗ.