Обработка списков на Prolog

Задание

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

1. Найти длину списка (по верхнему уровню);
2. Найти сумму элементов числового списка
3. Найти сумму элементов числового списка, стоящих на нечетных позициях исходного списка (нумерация от 0)

Убедиться в правильности результатов. Для одного из вариантов ВОПРОСА и одного из заданий составить таблицу, отражающую конкретный порядок работы системы

Код программы

```
include "lab17.inc"

domains
  intlist = integer*

predicates
  recursion_length(integer, integer, intlist)
  length(integer, intlist)

  recursion_sum(integer, integer, intlist)
  sum(integer, intlist)

  recursion_oddsum(integer, integer, intlist)
  oddsum(integer, intlist)

clauses
  % 1.
  recursion_length(Res, Len, [_ | Tail]) :- NewLen = Len + 1, !,
recursion_length(Res, NewLen, Tail).
  recursion_length(Res, Len, []) :- Res = Len.
  length(Res, List) :- recursion_length(Res, 0, List).
  % 2.
  recursion_sum(Res, Sum, [Head | Tail]) :- NewSum = Sum + Head, !,
recursion_sum(Res, NewSum, Tail).
  recursion_sum(Res, Sum, []) :- Res = Sum.
  sum(Res, List) :- recursion_sum(Res, 0, List).
  % 3.
  recursion_oddsum(Res, Sum, [_, Head | Tail]) :- NewSum = Sum + Head, !,
recursion_oddsum(Res, NewSum, Tail).
  recursion_oddsum(Res, Sum, []) :- Res = Sum.
  oddsum(Res, List) :- recursion_oddsum(Res, 0, List).
```

```
goal
  %length(Res, [1, 2, 3, 4]).
        %Res=4
        %1 Solution
  %sum(Res, [1, 2, 3, 4]).
        %Res=10
        %1 Solution
  oddsum(Res, [1, 2, 3, 4]).
        %Res=6
        %1 Solution
```

Словесное описание порядка поиска ответа на вопрос

№ шага	**Состояние резольвенты, и вывод: дальнейшие действия (почему?)**	**Для каких термов запускается алгоритм унификации: T1=T2 и каков результат (и подстановка)**	**Дальнейшие действия: прямой ход или откат (почему и к чему приводит?)**
1	length(Res, [1, 2, 3, 4]).	Сравнение: length(Res, [1, 2, 3, 4]) = recursion_length(Res, Len, [_ | Tail]) Унификация неуспешна (несовпадение функторов)	Прямой ход, переход к следующему предложению
2	...	...	...
3	recursion_length(Res, 0, [1, 2, 3, 4])	Сравнение: length(Res, [1, 2, 3, 4]) = length(Res, List). Унификация успешна. Подстановка: {Res=Res, List=[1, 2, 3, 4]}	Прямой ход, редукция резольвенты
4	NewLen = 0 + 1 ! recursion_length(Res, NewLen, [2, 3, 4])	Сравнение: recursion_length(Res, 0, [1, 2, 3, 4]) = recursion_length(Res, Len, [_ | Tail]). Унификация успешна. Подстановка: {Res=Res, Len = 0, Tail=[2, 3, 4]}	Прямой ход, редукция резольвенты
5	! recursion_length(Res, 1, [2, 3, 4])	NewLen = 0 + 1. Подстановка: {..., NewLen=1}	Прямой ход, редукция резольвенты
6	recursion_length(Res, 1, [2, 3, 4])	!. Отсечение 4, 5	Прямой ход, редукция резольвенты
7	NewLen = 1 + 1 ! recursion_length(Res, NewLen, [3, 4])	Сравнение: recursion_length(Res, 1, [2, 3, 4]) = recursion_length(Res, Len, [_ | Tail]). Унификация успешна. Подстановка: {Res=Res, Len = 1, Tail=[3, 4]}	Прямой ход, редукция резольвенты
8	! recursion_length(Res, 2, [3, 4])	NewLen = 1 + 1. Подстановка: {..., NewLen=2}	Прямой ход, редукция резольвенты

№ шага	Состояние резольвенты, и вывод: дальнейшие действия (почему?)	Для каких термов запускается алгоритм унификации: T1=T2 и каков результат (и подстановка)	Дальнейшие действия: прямой ход или откат (почему и к чему приводит?)
9	recursion_length(Res, 2, [3, 4])	!. Отсечение 7, 8.	Прямой ход, редукция резольвенты
10	NewLen = 2 + 1 ! recursion_length(Res, NewLen, [4])	Сравнение: recursion_length(Res, 2, [3, 4]) = recursion_length(Res, Len, [_ | Tail]). Унификация успешна. Подстановка: {Res=Res, Len = 2, Tail=[4]}	Прямой ход, редукция резольвенты
11	! recursion_length(Res, 3, [4])	NewLen = 2 + 1. Подстановка: {..., NewLen=3}	Прямой ход, редукция резольвенты
12	recursion_length(Res, 3, [4])	!. Отсечение 10, 11	Прямой ход, редукция резольвенты
13	NewLen = 3 + 1 ! recursion_length(Res, NewLen, [])	Сравнение: recursion_length(Res, 3, [4]) = recursion_length(Res, Len, [_ | Tail]). Унификация успешна. Подстановка: {Res=Res, Len = 3, Tail=[]}	Прямой ход, редукция резольвенты
14	! recursion_length(Res, 4, [])	NewLen = 3 + 1. Подстановка: {..., NewLen=4}	Прямой ход, редукция резольвенты
15	recursion_length(Res, 4, [])	!, отсечение 13, 14	Прямой ход, редукция резольвенты
16		Сравнение: recursion_length(Res, 4, []) = recursion_length(Res, Len, [_ | Tail]). Унификация неуспешна	Прямой ход, следующее предложение
17	Res = 4	Сравнение: recursion_length(Res, 4, []) = recursion_length(Res, Len, []). Унификация успешна. Подстановка: {Res=Res, Len = 4}	Прямой ход, редукция резольвенты
18		Res = 4. Подстановка: {..., Res=4}	**Вывод: Res=4**
19	...	...	...