**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, искусственный интелект и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

# Отчёт по лабораторной работе № 11 по курсу «Функциональное и логическое программирование»

**Тема** Рекурсия на Prolog

**Студент** Калашков П. А.

**Группа** ИУ7-66Б

**Оценка (баллы)**

**Преподаватели** Толпинская Н. Б., Строганов Ю. В.

Москва — 2023 г.

# Задание

Используя хвостовую рекурсию, разработать (комментируя назначение аргументов) эффективную программу, 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);
4. Сформировать список из элементов числового списка, больших заданного значения;
5. Удалить заданный элемент из списка (один или все вхождения).
6. Объединить два списка.

Убедиться в правильности результатов.

Для одного из вариантов ВОПРОСА уметь составить таблицу, отражающую конкретный порядок работы системы (№ шага, состояние резольвенты, для каких термов запускаетя алгоритм унификации, дальнейшие действия: прямой ход или откат).

# Решение

Листинг 1 – Листинг программы

```
domains
  num = integer.
  list = num*.

predicates
  len(list, integer).
  len(list, integer, integer).

  sum(list, integer).
  sum(list, integer, integer).

  sumOdd(list, integer).
  sumOdd(list, integer, integer).

  biggerThan(list, integer, list).

  remove(list, num, list).

  myConcat(list, list, list).

clauses
  len(List, Result) :- len(List, 0, Result).
  len([], Result, Result) :- !.
  len([_|T], Cur, Result) :-
    NextCur = Cur + 1,
    len(T, NextCur, Result).

  sum(List, Result) :- sum(List, 0, Result).
  sum([], Result, Result) :- !.
  sum([H|T], Cur, Result) :-
    NextCur = Cur + H,
    sum(T, NextCur, Result).

  sumOdd(List, Result) :- sumOdd(List, 0, Result).
  sumOdd([], Result, Result) :- !.
  sumOdd([_], Result, Result) :- !.
  sumOdd([_, H|[T]], Cur, Result) :-
```

```
38      NextCur = Cur + H,
39      sumOdd(T, NextCur, Result).
40
41    biggerThan([], _, []) :- !.
42    biggerThan([H|T], Border, [H|ResT]) :-
43      H > Border,
44      biggerThan(T, Border, ResT), !.
45    biggerThan([_|T], Border, Result) :-
46      biggerThan(T, Border, Result).
47
48    remove([], _, []) :- !.
49    remove([El|T], El, Result) :-
50      remove(T, El, Result), !.
51    remove([H|T], El, [H|ResT]) :-
52      remove(T, El, ResT).
53
54    myConcat([], List, List).
55    myConcat([H|T], List, [H|ResT]) :-
56      myConcat(T, List, ResT).
57
58  goal
59    %len([1, 2, 3], Result).
60    %sum([1, 2, 3], Result).
61    %sumOdd([1, 2, 3], Result).
62    %biggerThan([1, 2, 3, 4, 5, 6, 7, 8], 4, Result).
63    %remove([1, 2, 3, 4, 5, 6, 7, 8], 4, Result).
64    %myConcat([1, 2, 3], [4, 5, 6], Result).
```