**Студент: Красоткин Семён**
**Группа: М80-208Б-19**
**Вариант: 3**

# Лабораторная работа №1: Обработка списков

**Стандартные предикаты**

**1] Длина списка:**

***lengthSimon([], 0).***
***lengthSimon([_|Tail], Length) :- length(Tail, Length1), Length is Length1 + 1.***

**Для пустого списка длина равна нулю, иначе отсекается голова и рекурсивно ищется длина нового списка приплюсовывая к нему 1.**

**Протокол:**
**lengthSimon([1,2,3], X).**
X = 3.
**lengthSimon([], X).**
X = 0.
**lengthSimon([2,2,2,1,2,2,2,7], X).**
X = 8.
**lengthSimon([1,2,3], 3).**
true.
**lengthSimon([0,2], 6).**
False.

**2] Принадлежность элемента списку**

**memberSimon(element, list).**
**memberSimon(Element, [Element|_]).**
**memberSimon(Element, [_|Tail]) :- memberSimon(Element, Tail).**

**Если список начинается с нужного элемента, то он есть в списке; иначе следует отсечь голову и рекурсивно искать элемент в остатке списка.**

**Протокол:**
**memberSimon(1, [1,2,3]).**
true.
**memberSimon(2, [1,5,1,3]).**
false.
**memberSimon(X, [1,2,3]).**
X = 1
X = 2
X = 3
false

**3] Конкатенация списков:**
**appendSimon([], List, List).**
**appendSimon([Head|Tail], List2, [Head|ResList2]) :- appendSimon(Tail, List2, ResList2).**

**Протокол:**
**appendSimon([1,2,3], [3,2,1], X).**
X = [1, 2, 3, 3, 2, 1]
**appendSimon([], [], X).**
X = []
**appendSimon([1, 1], [1, 1, 1], X).**
X = [1, 1, 1, 1, 1]
**appendSimon(X, Y, [1, 2, 3]).**
X = [],
Y = [1, 2, 3]

**При конкатенации пустого списка с другим, результат будет второй; иначе, отсекая голову первого списка, добавляю её в результирующий и так далее.**

**4] Удаление из списка:**

**removeSimon(X, [X|T], T).**
**removeSimon(X, [H|T], [H|T1]) :- removeSimon(X, T, T1).**

**Протокол:**

**removeSimon(1, [1,2,3], X).**
X = [2, 3]
**removeSimon(1, [1,2,1,3,1,4,1], X).**
X = [2, 1, 3, 1, 4, 1]
**removeSimon(1, [2,3,4], X).**
false

**Если удаляемый элемент находится в голове, то ответ - список без головы, иначе отсекаю голову списка и пытаюсь удалить хвостовой элемент. В результате список без головы и списка Tail без элемента Element (список Tail1). Если удаляемого элемента нет, то выводится false.**

**5] Перестановка:**

**permuteSimon([], []).**
**permuteSimon(List1, [X|P]) :- removeSimon(X, List1, List2),**
**permuteSimon(List2, P).**

**Протокол:**

**permuteSimon([1,2,3], X).**
X = [1, 2, 3]
X = [1, 3, 2]
X = [2, 1, 3]
X = [2, 3, 1]
X = [3, 1, 2]
X = [3, 2, 1]
false
**permuteSimon([1,2,3,4], [4,1,2,3]).**
true
**permuteSimon([1,2,3,4], [1,5,2,3]).**
false

**Конец рекурсии: перестановка пустого списка это пустой список, а если нет, то надо удалить из него один элемент и рекурсивно выполнить перестановку полученного списка. Удалённый элемент добавить в голову новой перестановки.**

**6] Подсписок в списке**

```
sublistSimon([], _).
sublistSimon([H|T], [H|L1]) :-
   sublistSimon(T, L1).
sublistSimon([H|T], [_|L1]) :-
   sublistSimon([H|T], L1).
```

**Протокол:**

**sublistSimon([1,2], [3,4,1,2,6]).**
true
**sublistSimon([1,2,3], [1,4,2,3]).**
true
**sublistSimon(X, [1,2,3,4]).**
X = []
X = [1]
X = [1, 2]
X = [1, 2, 3]
X = [1, 2, 3, 4]
X = [1, 2, 4]
X = [1, 3]
X = [1, 3, 4]
X = [1, 4]
X = [2]
X = [2, 3]
X = [2, 3, 4]
X = [2, 4]
X = [3]
X = [3, 4]
X = [4]

**Список Sublist - подсписок списка, если есть подсписок, из начала которого можно выделить подсписок.**

## Предикат обработки списков

**Ревёрс через свои предикаты:**

```
reversePSimon([],[]).
reversePSimon([H|T],R) :-
reversePSimon(T,RevT),appendSimon(RevT,[H],R).
```

**Ревёрс не через свои предикаты:**

```
reverseAcc([H|T],A,ResX) :- reverseAcc(T,[H|A],ResX).
reverseAcc([],A,A).
reverseSSimon(X,ResX) :- reverseAcc(X,[],ResX).
```

## Предикат обработки числовых списков

```
% Максимум списка без стандартных предикатов
maxListSimon([X],X) :- !, true.
maxListSimon([X|Xs], M):- maxListSimon(Xs, M), M >= X.
maxListSimon([X|Xs], X):- maxListSimon(Xs, M), X >  M.
```