# Задание №21
# Обработка списков

## 1. Общая постановка задачи

Для задания, соответствующего номеру вашего варианта, опишите на языке Prolog предикат, выводящий на экран результат для заданных аргументов.

Приведите примеры запросов, демонстрирующие всестороннее использование вашего предиката.

Опишите программу в текстовом файле с именем `task21-NN.pro`, где `NN` — номер вашего варианта. Полученный файл загрузите на портал в качестве выполненного задания.

## Пример выполнения задания

ТЕКСТ ЗАДАЧИ: *Описать функцию, удаляющую подряд идущие дубликаты элементов в заданном списке.*

РЕШЕНИЕ: Содержимое файла `task21-NN.pro`:

```
delDuplicates([], []).
delDuplicates([X], [X]).
delDuplicates([X, Y | L], [X | R]) :-
      X \= Y, delDuplicates([Y | L], R).
delDuplicates([X, X | L], R) :- delDuplicates([X | L], R).

delDuplicates(X) :- delDuplicates(X, Res), write(Res).

?- delDuplicates([ 111, 111, 111, 2, 2, 2, 2
                 , 3, 3, 3, 2, 2, 2, 1
                 , 1, 1, 2, 3, 3, 43, 4
                 , 5, 6, 6, 6, 6, 4, 4, 4
                 , 4, 4, 4, 3, 3, 3, 2]).
```

## 2. Предварительные замечания

Все решения должны быть представлены в виде алгоритма с хвостовой рекурсией.

Возможно определение любого количества вспомогательных предикатов.

При необходимости использования механизмов `reverse`, `memeber` и т.п. должна быть проведена самостоятельная реализация этих механизмов.

Не забывайте, что решение должно быть оформлено в соответствии с правилами. Не соответствие правилам оформления влечет штраф 20

Решение может быть засчитано только если задание выполнено на 100%.

Не следует делать предположений на счет задания, не сформулированных явно в условии. Если возникают сомнения — задайте вопрос на форуме «Язык Prolog».

## Варианты заданий

**1.** Опишите предикат `palindrome(L)`, определяющий является ли заданный список `L` палиндромом (инвертированный список равен исходному).

**2.** Опишите предикат `mucSum(L)`, который из исходного списка формирует список-результат: первый элемент — сумма всех элементов, второй — сумма элементов хвоста, третий — сумма элементов от третьего до последнего, и т. д..

**3.** Опишите предикат `alternate(L)`, получающий список целых чисел и подсчитывающий сумму его элементов с переменой знака. Например, вызов `alternate([1, 2, 3, 4])` должен приводить к получению
`1 - 2 + 3 - 4 = -2`.

**4.** Опишите предикат `scalar(L1, L2)`, который вычисляет скалярное произведение (сумму произведений соответствующих элементов) двух числовых списков (результат — число). Если во входных списках задано разное количество элементов, то скалярное произведение должно вычисляться для первых $N$ элементов, где $N$ — количество элементов в более коротком списке.

**5.** Реализовать предикат `insertEl(L ,A, N)`, включающий объект `A` на позицию `N` в списке `L`. Позиция в списке задается порядковым номером элемента, отсчитываемым от 0. Если номер заданной позиции больше чем количество элементов в данном списке, то последний элемент списка должен быть продублирован недостающее количество раз.

**6.** Реализовать предикат `intersection(M1, M2)` для получения пересечения двух множеств `M1` и `M2`, где каждое множество представляется списком неповторяющихся элементов.

**7.** Опишите предикат `isSubset(M1, M2)`, определяющий, является ли первое множество подмножеством второго. Каждое множество задается в виде списка неповторяющихся элементов.

**8.** Опишите предикат `sameOrder(L1, L2)`, возвращающий `true`, если все элементы, входящие в оба списка, расположены в этих списках в одном и том же порядке. Считаем, что в каждом списке нет повторяющихся элементов.

**9.** Описать предикат `backPos(L1, N)`, выдающий элемент списка `L1` по заданному номеру `N`, отсчитывая элементы от конца списка.

**10.** Опишите предикат `delKth(L, K)`, удаляющий из списка `L` каждый `K`-ый элемент. К примеру, из `[3, 1, 3, 2, 3, 5, 4, 4, 7, 9, 6, 6]` при `K` равном 4 должен получиться список `[3, 1, 3, 3, 5, 4, 7, 9, 6]`.

**11.** Опишите предикат `insertSublist(L1, L2, K)`, который в заданный список `L1` вставлял бы с данной позиции `K` элементы второго списка `L2`.

**12.** Опишите предикат `sustr(L1, L2)`, принимающий в качестве аргументов два списка и возвращающий в качестве результата список, содержащий элементы первого списка, не принадлежащие второму списку.

**13.** Опишите предикат `repeat(L1, L2)`, получающий два списка: список целых чисел и список неотрицательных целых чисел. Предикат должен в списке-результате повторять каждый элемент первого списка то количество раз, которое указано в соответствующем элементе второго списка. Например, вызов
`repeat ([1, 2, 3], [4, 0, 3])` должен приводить к получению `[1, 1, 1, 1, 3, 3, 3]`.
Если количество элементов в первом и втором списке не совпадает, то необходимо обрабатывать минимальное количество из предложенных элементов списка.

**14.** Опишите предикат `zipRecycle(L1, L2)`, получающий два непустых числовых списка и возвращающий список пар, в котором каждая $i$-я пара содержит $i$-е элементы заданных списков. Если длина заданных списков отличается, то в результирующем списке должно быть столько же элементов, сколько в списке с максимальной длиной, причем элементы списка с минимальной длиной должны циклически повторяться. Например, запуск
`zipRecycle([1, 2, 3], [1, 2, 3, 4, 5, 6, 7])`
должен приводить к получению
`[(1, 1), (2, 2), (3, 3), (1, 4), (2, 5), (3, 6), (1, 7)]`.

**15.** Опишите предикат `splitAt(L, N)`, получающий числовой список и число. Предикат должен выводить два списка, где первый содержит все элементы заданного списка, превышающие по величине второй аргумент, а второй — все остальные элементы. Порядок элементов в списках должен сохраниться прежним.

**16.** Опишите предикат `isAnySorted(L)`, выдающий `true`, если заданный список отсортирован по убыванию или возрастанию.

**17.** Опишите предикат `maxSorted(L)`, принимающий список и выводящий на экран максимальную по длине последовательность соседних элементов, упорядоченных по возрастанию. Если таких последовательностей несколько, то необходимо вернуть первую из них. Например, вызов
`maxSorted([3, 1, 3, 2, 3, 5, 4, 4, 7, 9, 6])` должен привести к получению результата `[2, 3, 5]`.

**18.** Опишите предикат `sortedMerge(L1, L2)`, получающий два отсортированных в порядке возрастания числовых списка и выводящий объединенный список всех элементов, отсортированный в том же порядке. Например, вызов `sortedMerge ([1, 4, 7], [5, 8, 9])` должен приводить к выводу `[1, 4, 5, 7, 8, 9]`.