Zadanie 9.1

W pliku **listy.pl** zdefiniuj predykat **take/3** pobierający N początkowych elementów z listy. Argumenty:

- liczba naturalna,
- lista początkowa,
- lista wynikowa.

Uwaga: Należy rozpatrzeć dwa warunki końcowe: pobieranie z pustej listy i pobieranie „0" elementów.

Zadanie 9.2

(*) W pliku **listy.pl** zdefiniuj predykat **drop/3** usuwający N początkowych elementów z listy. Argumenty:

- liczba naturalna,
- lista początkowa,
- lista wynikowa.

Uwaga: Należy rozpatrzeć dwa warunki końcowe: usuwanie z pustej listy i usuwanie „0" elementów.

Zadanie 9.3

W pliku **listy.pl** zdefiniuj predykat **init/2** pobierający z listy L wszystkie elementy poza ostatnim. Pierwszym argumentem predykatu jest lista wejściowa, a drugim wynik. Podaj dwie definicje tego predykatu:

1. z zastosowaniem predykatu **reverse**;
2. (*) z zastosowaniem predykatu **take**.

Zadanie 9.4

W pliku **listy.pl** zdefiniuj predykat **middle/2** pobierający z listy L wszystkie elementy poza pierwszym i ostatnim. Pierwszym argumentem predykatu jest lista wejściowa, a drugim wynik.

Zadanie 9.5

(*) W pliku **listy.pl** zdefiniuj predykat **split/3**, który dzieli listę L na dwie części L1 i L2 mniej więcej równej długości (z dokładnością do jednego elementu).

Zadanie 9.6

Dodaj predykat **permutation/2** do pliku **listy.pl** i przeanalizuj jego działanie. Przetestuj działanie predykatu na przykładach.

```
% permutation(P, L) -  P jest permutacją listy L


permutation([], []).


permutation(P, [H|T]) :-
    permutation(P1, T),
```

```
    append(A, B, P1),       % P1 dzielimy na dwie części
    append(A, [H|B], P).    % P powstaje poprzez wstawienie
                            % elementu H w pewne miejsce listy P1
```

Zadanie 9.7

(*) W pliku **listy.pl** zdefiniuj predykat **middle/3** pobierający z listy L wszystkie elementy z pominięciem N pierwszych i N ostatnich. Pierwszym argumentem predykatu jest liczba odcinanych z każdej strony listy elementów, drugim jest lista wejściowa, a trzecim wynik.

Uwaga: Z listy można odcinać elementy jeżeli jej długość jest większa niż 2 * N. W przeciwnym przypadku wynikiem jest lista pusta.

Należy skorzystać z predykatów **take/2** i **drop/2**.

Zadanie 9.8

(*) W pliku **listy.pl** zdefiniuj predykat **move/2** przesuwający cyklicznie o 1 elementy z listy, np.:

```
?- move([1,2,3,4], X), move(X, Y), move(Y, Z).
X = [2, 3, 4, 1],
Y = [3, 4, 1, 2],
Z = [4, 1, 2, 3]
```

Zadanie 9.9

W pliku **analiza.pl** zdefiniuj predykaty wyliczające poniższe sumy i iloczyny:

1. suma od 1 do 100 liczb postaci 1/n;
2. iloczyn od 1 do 50 liczb postaci (1+n)/(2+n);
3. (*) suma od 1 do 1000 liczb postaci 1/(i^2);
4. (*) suma od 1 do 1000 liczb postaci (sqrt i)-1/i;
5. (*) iloczyn od 1 do 1000 liczb postaci (i+1)/(i^3).

Uwaga: Do pliku **analiza.pl** skopiuj predykat **listSum** ze slajdu 41. Zobacz przykłady ze slajdu 47.