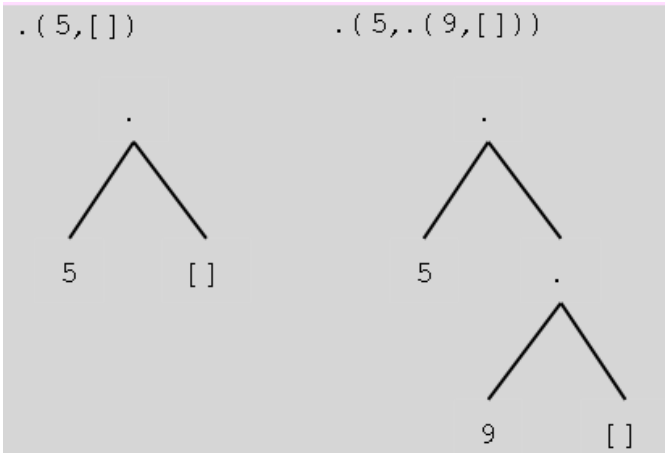
Praca z listami w Prologu

Lista jest podstawową strukturą rekurencyjną w języku Prolog. Jest ona ciągiem złożonym z dowolnej liczby elementów, którymi mogą być termy tj. stałe, zmienne i struktury. Jako że lista jest strukturą (termem złożonym), do jej konstrukcji użyto dwuargumentowego funktora . (tzw. operator cons). Poniżej podano przykłady list zapisanych przy pomocy tego operatora.



(Ogólnie dla listy zawierającej n elementów należy użyć operatora . n razy. Koniec listy jest zaznaczany przy użyciu specjalnego atomu [], który oznacza też listę pustą.)
Aby uniknąć uciążliwego zapisu za pomocą funktora ., przyjęto zamiast niego używać nawiasów kwadratowych, zaś elementy listy odzielać przecinkami.
Każda lista składa się z:

- **głowy** (ang. head), która jest zawsze 1. elementem listy, oraz
- **ogona** (ang. tail), który jest zawsze listą

Listę zapisujemy:

```
[a,b,c]
[2,4,6,ala,ma,kota]
[]
```

Przykład podziału listy na głowę i ogon:

lista	głowa	ogon
[a,b,c,d]	a	[b,c,d]
[a,b,[c,d]]	a	[b,[c,d]]
[]	*brak*	*brak*
[elemencik]	elemencik	[]
[[1,2],[3,4],5]	[1,2]	[[3,4],5]

Zadanie 1.

Sprawdzić wyniki, jakie daje SWI-Prolog dla następujących zapytań (nie trzeba wprowadzać do bazy danych żadnych faktów):

```
?- [ala,  ma, kota] = [kota, ma, ala].
?- [1,2,3] = [X,Y].
?- [1,2,3,osiem] = [1|Ogon].
?- [1|[2|[3|[osiem]]]] = [1|Ogon].
?- [[0,1,2],[3,4],[5]] = [Glowa|Ogon].
?- [ala,ma,kota,a,ola,ma,psa] = [ala,ma,kota,a|X].
?- [alfa(1,2), alfa(3,4), alfa(5,6)] = [alfa(X,2)|Ogon].
```

Zadanie 2.
Zdefiniować predykat pisz_listę(L), który wypisuje na ekranie wszystkie elementy listy L. Skorzystać z predykatu jednoargumentowego write. Wskazówka: napisać dwie reguły dla predykatu pisz_listę(X) - (1) wypisanie listy pustej, (2) wypisanie listy, składającej się z głowy i ogona (z wywołaniem rekurencyjnym).

Zadanie 3.
Zdefiniować predykat należy(E, L) - element E należy do listy L. Wskazówka: sformułować dwie reguły - (1) E jest głową listy, (2) E należy do ogona listy.

Zadanie 4.
Zdefiniuj predykat wyznaczający długość listy.

Zadanie 5.
Zdefiniuj predykat pozwalający na łączenie dwóch list.

Zadanie 6.
Zdefiniuj predykat pozwalający na dodawanie elementów do listy.

Zadanie 7.
Zdefiniuj predykat pozwalający na usuwanie elementów do listy.

Zadanie 8.
Znaleźć ostatni element listy (zdefiniować predykat ostatni(E, L)).

Zadanie 9.
Sprawdzić, czy pierwsza lista jest początkiem drugiej listy (zdefiniować predykat początek(Lista1,Lista2)).

Zadanie 10.
Zdefiniować predykat dopasuj(L, W, Z), który dla listy *L* zwraca jej podlistę *Z* pasującą do zadanego wzorca *W*. Wzorzec jest listą, której elementami mogą być następujące stałe

- n - pasuje do liczby,
- a - pasuje do atomu,
- l - pasuje do elementu będącego (dowolną) listą,
- . - pasuje do dowolnego elementu,
- *- pasuje do podlisty o dowolnej długości (także pustej).

Np. wzorzec [n,n] oznacza podlistę złożoną z dwóch liczb, zaś wzorzec [a,*,n] - podlistę zaczynającą się atomem i kończącą się liczbą.
Wskazówka: użyć predykatów wbudowanych atom, integer.
Przykład oczekiwanego działania predykatu:

```
?- dopasuj([x,a,15,101,ala,ma,kota,[1,2],a,b,c],[n,n,*,l,.],Z).
Z = [15, 101, ala, ma, kota, [1,2], a]
Yes
```