

Metody programowania 2017

Lista zadań nr 2

Na zajęcia 7 i 8 marca 2017

Zadanie 1 (1 pkt). Przypomnijmy predykat `append/3`:

```
append([], X, X).  
append([H|T], X, [H|Y]) :-  
    append(T, X, Y).
```

Narysuj prologowe drzewo poszukiwań dla celu:

```
?- append(X, Y, [a,b,c]).
```

Zadanie 2 (1 pkt). Przypomnijmy predykat `select/3`:

```
select(H, [H|T], T).  
select(X, [H|T], [H|S]) :-  
    select(X, T, S).
```

Narysuj prologowe drzewo poszukiwań dla celu:

```
?- select(x, L, [a,b,c]).
```

Zadanie 3 (1 pkt). Zadać Prologowi następujące pytania:

1. Jakie pary list mają tę własność, że druga jest wynikiem konkatencji dwóch kopii pierwszej z nich?
2. Jaki element należy usunąć z listy `[a,b,c,d]` by otrzymać listę `[a,c,d]`?
3. Jaką listę należy dołączyć do listy `[a,b,c]`, by otrzymać listę `[a,b,c,d,e]`?

Zadanie 4 (1 pkt). Zaprogramuj w Prologu poniższe predykaty:

1. `even(X)`, spełniony gdy lista `X` ma parzystą liczbę elementów;
2. `palindrom(X)`, spełniony gdy lista `X` jest palindromem;
3. `singleton(L)`, spełniony gdy `X` jest listą jednoelementową.

Zadanie 5 (1 pkt). Zaprogramuj w Prologu poniższe predykaty:

1. `head(H,L)`, spełniony gdy `H` jest pierwszym elementem (głową) listy `L`;
2. `last(L,H)`, spełniony gdy `H` jest ostatnim elementem listy `L`;
3. `tail(T,L)`, spełniony gdy `T` jest listą wszystkich z wyjątkiem pierwszego elementów (ogonem) listy `L`;
4. `init(L,T)`, spełniony gdy `T` jest listą wszystkich z wyjątkiem ostatniego elementów listy `L`;
5. `prefix(P,L)`, spełniony gdy `P` jest listą początkowych elementów (prefiksem) listy `L`;
6. `suffix(L,S)`, spełniony gdy `S` jest listą końcowych elementów (sufiksem) listy `L`.

Zadanie 6 (1 pkt). Zaprogramuj w Prologu taki predykat `sublist/2`, że cel

`sublist(l1, l2)`

jest spełniony, gdy lista `l2` jest *podlistą* listy `l1`. Przy kolejnych nawrotach predykat powinien wygenerować wszystkie podlisty w dowolnej kolejności. Podlistą listy `[x1, ..., xn]` nazywamy dowolną listę postaci `[xi1, ..., xik]` dla $1 \leq i_1 < \dots < i_k \leq n$. Podlista powstaje zatem przez usunięcie z listy niektórych elementów. Lista n -elementowa ma 2^n podlist.

Zadanie 7 (1 pkt). *Permutacją* listy `[x1, ..., xn]` nazywamy dowolną listę postaci

`[xi1, ..., xin]`,

gdzie i_1, \dots, i_n są parami różnymi liczbami z przedziału $1, \dots, n$. Permutacja powstaje zatem przez przestawienie kolejności elementów na liście. Lista n -elementowa ma $n!$ permutacji.

Permutacje można generować *przez wybieranie*, zgodnie z następującym schematem rekurencyjnym:

- Jedyną permutacją listy pustej jest lista pusta.
- Aby wygenerować permutację pewnej niepustej listy, wybierz z niej dowolny element. Wybrany element będzie głową tworzonej permutacji. Jej ogonem będzie natomiast permutacja listy pozostałej po wybraniu tego elementu.

Zaprogramuj predykat `perm/2` implementujący powyższy algorytm.