Metody programowania 2017

Lista zadań nr 3

Na zajęcia 14 i 15 marca 2017

Zadanie 1 (1 pkt). Permutacje (rozważane w zadaniu 7 z poprzedniej listy) można także generować *przez wstawianie*, zgodnie z następującym schematem rekurencyjnym:

- Jedyną permutacją listy pustej jest lista pusta.
- Aby wygenerować permutację pewnej niepustej listy, wygeneruj dowolną permutację jej ogona i następnie wstaw jej głowę w dowolne miejsce w wygenerowanej wcześniej permutacji.

Zaprogramuj predykat `perm/2` implementujący powyższy algorytm.

Zadanie 2 (1 pkt). Zaprogramuj w Prologu predykaty:

1. `filter(+LNum, ?LPos)`, spełniony, gdy `LPos` unifikuje się z podlistą listy `LNum` zawierającą wszystkie nieujemne elementy listy `LNum`.
2. `count(+Elem, +List, ?Count)`, spełniony, gdy `Elem` unifikuje się z dokładnie n elementami listy `List` i `Count` unifikuje się z liczbą n.
3. `exp(+Base, +Exp, ?Res)`, spełniony, gdy `Res` unifikuje się wynikiem podniesienia liczby `Base` do potęgi `Exp`.

Zadanie 3 (1 pkt). Zaprogramuj w Prologu predykaty:

1. `factorial(+N, ?M)` spełniony, gdy `M` unifikuje się z silnią liczby `N`.
2. `concat_number(+Digits, ?Num)`, spełniony, gdy lista `Digits` zawiera ciąg cyfr rozwinięcia dziesiętnego liczby, która unifikuje się z `Num`.
3. `decimal(+Num, ?Digits)`, spełniony, gdy `Digits` unifikuje się z z ciągiem cyfr rozwinięcia dziesiętnego liczby `Num`. Program ma zwracać poprawny wynik dla liczb nieujemnych.

Zadanie 4 (1 pkt). Zaprogramuj predykat

```
select_min(+NumList, ?Min, ?Rest)
```

który wybiera najmniejszy element `Min` listy liczb `NumList` i zwraca pozostałe elementy na liście `Rest`. Wykorzystaj go do zaprogramowania predykatu `sel_sort/2` sortującego listę liczb całkowitych używając algorytmu sortowania przez wybieranie.

Zadanie 5 (1 pkt). Zaprogramuj predykat

```
insert(+NumList, +Elem, ?Res)
```

który wstawia, zachowując porządek, liczbę `Elem` do listy liczb całkowitych `NumList` i unifikuje wynik z `Res`. Wykorzystaj go do zaprogramowania predykatu `ins_sort/2` sortującego listę liczb całkowitych za pomocą algorytmu sortowania przez wstawianie.

Zadanie 6 (1 pkt). Najprostsza implementacja predykatu `reverse/2`:

```
reverse(X,Y) :-
   reverse(X,[],Y).

reverse([],A,A).
reverse([H|T],A,Y) :-
   reverse(T,[H|A],Y).
```

zapętla się w trybie `(-,+)`, tj. gdy pierwszy argument jest nieukonkretniony, a drugi ukonkretniony. Relacja „jest odwróceniem" jest symetryczna, oczekiwalibyśmy więc, że obliczenie celów `reverse`(a, b) oraz `reverse`(b, a) powinno dawać dokładnie ten sam efekt. Zaprogramuj predykat `reverse/2` w taki sposób, by nie zapętlał się dla żadnych danych.

Zadanie 7 (1 pkt). Relacja „być permutacją" także jest symetryczna. Niestety w trybie `(-,+)` implementacje predykatu `perm/2` rozważane na poprzedniej i obecnej liście zadań również nie działają zgodnie z intuicją. Popraw je tak, by wywołania `perm`(a, b) oraz `perm`(b, a) zawsze dawały ten sam efekt.