Programowanie 2010

Lista zadań nr 3

Na zajęcia 16–18 marca 2010

Zadanie 1 (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 2 (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 3 (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 4 (1 pkt). Zaprogramuj predykat `insert(+NumList, +Elem, ?Res)`, który wstawia liczbę `Elem` do listy liczb całkowitych `NumList` nie zaburzając porządku i unifikuje wynik z `Res`. Wykorzystaj go do zaprogramowania predykatu `ins_sort`/2 sortującego listę liczb całkowitych za pomocą algorytmu sortowania przez wstawianie.

Programming 2010

Problem set no. 3

Due March 16, 2010

Problem 1 (1 p). Define the following predicates in Prolog:

1. `filter(+LNum, ?LPos)`, satisfied when `LPos` unifies with a sublist of the list `LNum` that contains all nonnegative elements of the list `LNum`.
2. `count(+Elem, +List, ?Count)` satisfied when `Elem` unifies with exactly n elements of the list `List` and `Count` unifies with the number n.
3. `exp(+Base, +Exp, ?Res)` satisfied when `Res` unifies with the result of taking the number `Base` to the power of `Exp`.

Problem 2 (1 p). Define the following predicates in Prolog:

1. `factorial(+N, ?M)` satisfied when `M` unifies with the factorial of `N`.
2. `concat_number(+Digits, ?Num)` satisfied when the list `Digits` contains a sequence of digits of the decimal representation of the number that unifies with `Num`.
3. `decimal(+Num, ?Digits)` satisfied when `Digits` unifies with the list of digits of the decimal representation of the number `Num`. Your program should return the correct result for nonnegative numbers.

Problem 3 (1 p). Define a predicate `select_min(+NumList, ?Min, ?Rest)` that selects the smallest element `Min` from the list `NumList` of integers and returns the remining elements in `Rest`. Use it to define a predicate `sel_sort`/2 which sorts a list of integers using the selection sort algorithm.

Problem 4 (1 p). Define a predicate `insert(+NumList, +Elem, ?Res)` which inserts the number `Elem` into the list `NumList` preserving the ordering of elements and unifies the result with `Res`. Use it to define a predicate `ins_sort`/2 which sorts a list of integers using the insertion sort algorithm.

Zadanie 5 (1 pkt). Najprostsza implementacja predykatu `reverse`/2:

Problem 5 (1 p). The simplest implementation of the `reverse`/2 predicate:

```
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.

loops when used in `(-,+)` mode, *i.e.* when its first argument is not instantiated while the second one is. Relation "is the reversion of" is symmetric, so we would rather expect the computation of goals `reverse`(a, b) and `reverse`(b, a) to give exactly the same outcome. Write the `reverse`/2 predicate in such a way that it never loops for any input data.

Zadanie 6 (1 pkt). Relacja „być permutacją" także jest symetryczna. Niestety w trybie `(-,+)` implementacje predykatu `perm`/2 z zadania 1 również nie działają zgodnie z intuicją. Popraw je tak, by wywołania `perm`(a, b) oraz `perm`(b, a) zawsze dawały ten sam efekt.

Problem 6 (1 p). The "to be a permutation" relation is symmetric as well. Unfortunately both implementations of the predicate `perm`/2 from the Problem 1 also work contrary to our intuition when called in the `(-,+)` mode. Correct them so that any pair of calls `perm`(a, b) and `perm`(b, a) gives the same outcome.