Programowanie logiczne
Pracownia 1
Termin: Zajęcia nr 2

Zadania występujące na tej liście są przeznaczone do automatycznego sprawdzania, ale wagi dla efektywności oraz stylu są równe 0 (lub prawie 0). Oczywiście pewne aspekty rozwiązań (niebąbelkowość algorytmu sortowania, czy też różność wariantów predykatu `perm` wymagają sprawdzenia „ręcznego".

Zadania z zerową liczbą punktów nie będą sprawdzane automatycznie. Zachęcam wszystkich (zwłaszcza osoby, dla których to jest pierwszy kontakt z Prologiem) do rozwiązywania tych zadań i ewentualnych konsultacji z prowadzącym zajęcia.

Zadanie (0p). Zdefiniuj na liczbach w reprezentacji unarnej[1] relacje:

a) `exp(X,Y,Z)` oznaczającą $\mathtt{X}^{\mathtt{Y}} = \mathtt{Z}$,

b) `minus(X,Y,Z)` oznaczającą $\max(0, \mathtt{X} - \mathtt{Y}) = \mathtt{Z}$,

c) `mod(X,Y,Z)` oznaczającą $\mathtt{X} \bmod \mathtt{Y} = \mathtt{Z}$.

Zadanie (0p). Napisz predykat `select(X,L1,L2)` prawdziwy, gdy lista `L2` powstaje przez wstawienie do listy `L1` elementu `X`.

Zadanie (0p). 2 Napisz dwie, realizujące różne algorytmy, wersje predykatu `perm(X,Y)`. Predykat ten jest prawdziwy, gdy lista `Y` jest permutacją listy `X`. Predykat powinien nadawać się nie tylko do sprawdzania, ale również do generowania permutacji.

Zadanie to należy rozwiązać pisząc dwa predykaty: `perm1` oraz `perm2`

Zadanie 1.(1pkt) Napisz ogonową wersję predykatu sumującego wyrazy na liście liczb.

Zadanie 2.(1pkt) Napisz predykat `occurences(S,T,N)` prawdziwy, gdy `S` występuje w termie `T` dokładnie `N` razy. Jak można używać tego predykatu?

Zadanie 3.(1pkt) Wybierz swój ulubiony algorytm sortowania (np. `quicksort` albo `mergesort`, ale nie `bubblesort`) i zaimplementuj go w Prologu.

Zadanie 4.(2pkt) Sygnaturą nazwiemy listę zawierającą termy postaci `a/N`, mówiące o dopuszczalnych konstruktorach termów wraz z ich arnością. Przykładowa sygnatura to `[f/2,c/0,g/1]`, termem zbudowanym nad nią jest `f(c,g(f(c,c,))` . Wielkością termu nazwiemy liczbę wystąpień konstruktorów termów (wliczając w to również stałe, jako konstruktory 0-arne). Napisz predykat `term(Signature, Size, Term)`, prawdziwy, gdy `Term` ma rozmiar `Size` i zbudowany jest nad sygnaturą `Signature`. Dozwolone sposoby użycia to: `(+,?,+)` oraz `(+,+,?)` (czyli predykta powinien służyć zarówno do sprawdzania, czy term ma sygnaturę, do obliczanie wielkości termu i do generowania termu o zadanej sygnaturze i rozmiarze.

[1] W reprezentacji unarnej mamy stałą oznaczającą zero (0) oraz konstruktor oznaczający następnika, zatem przykładowo 4 zapisujemy jako s(s(s(s(0)))).

Zadanie (0p). Będziemy rozważać multizbiory, reprezentowane za pomocą atomu `void` i konstruktora `bag(Element, Krotność, ResztaMultizbioru)`. Reprezentacja multizbioru nazwiemy nienadmiarową, jeżeli każdy element wyzstępje w niej dokładnie raz. Napisz program znajdujący sumę oraz przecięcie dwóch multizbiorów.

Zadanie 5.(2pkt) Napisz program konwertujący listę elementów do multizbioru reprezentowanego nienadmiarowo.