Programowanie logiczne
Pracownia 2
Zajęcia 3 i 4
Sprawdzaczka pojawi się pod koniec tygodnia

Zadanie (0p). Załóżmy, że graf (skierowany, acykliczny) zadany jest relacją `edge(W1,W2)`. Napisz możliwie prosty program `connected(X,Y)` sprawdzający, czy istnieje ścieżka pomiędzy węzłami `X` oraz `Y` w tym grafie.

Zadanie 1.(1pkt) Napisz zdefiniowany podobnie jak w poprzednim zadaniu predykat `connected` niewymagający od grafu, by był acykliczny.

Zadanie 2.(7pkt) Napisz predykat, `simp(E,ES)`, który dla wyrażenia algebraicznego złożonego ze zmiennych (oznaczanych małymi literami), liczb całkowitych i znaków `+`, `*`, `-` znajdzie prostsze równoważne mu wyrażenie arytmetyczne. Twój program powinien:

1. Wykonywać jak najwięcej obliczeń na liczbach, przykładowo `2*x*y*3` powinno być zamienione na `6*x*y`
2. Eliminować powtarzające się wyrażenia, przykładowo: `x+y+x` zamieniac na `2*x`, a `x * (y+2) + x*(z+1)` na `x*(z+y+3)`.
3. Umieć, gdy trzeba skorzystać z przemienności bądź łączności i zamienić `x*(y+2)+z*(2+y)` na `(x+z)*(y+2)`

Zadanie sprawdzane automatycznie. Szczegóły zostaną podane wkrótce!

Zadanie 3.(1pkt) Napisz ponownie[1] program konwertujący listę elementów do multizbioru reprezentowanego nienadmiarowo. Przypilnuj, by program działał efektywnie.

Zadanie 4.(2pkt) Napisz predykat sortujący realizujący następujący algorytm: wybierz niedeterministycznie dwa sąsiednie elementy listy i jeżeli potrzeba, to je zamień. Powtarzaj ten proces, aż otrzymasz posortowaną listę.

Zadanie 5.(5pkt) Napisz program, który przetwarza listę reguł oraz listę faktów. Reguły przypominają prologowe klauzule i czytamy je jako implikacje. Mają one postać:

```
[a,b,...,c] >> [d,e, ..., f]
```

gdzie a,b,c,d,e,f są prologowymi atomami. Zakładamy ponadto, że fakty nie zawierają zmiennych, a w regułach każda zmienna po prawej stronie, występuje również po stronie lewej. Zastosowanie reguły polega na:

i) Utworzenie świeżej kopii reguły.

ii) Dopasowanie każdego atomu z przesłanek do któregoś z bierzącej listy faktów.

[1] Możesz również oddać poprzedni program, jeżeli uznasz, że nie da się go przyśpieszyć

iii) Dodanie konkluzji do bierzącej listy faktów.

Napisz predykat: `inference(Facts, Rules, MaximumFaktLength, Result)`, który przeprowadza powyższą procedurę tak długo, jak się da (to znaczy tak długo, jak generuje nowe fakty) i tworzy listę wydedukowanych faktów. Maksymalny rozmiar wydedukowanych faktów zapewnie, że powyższa procedura się skończy.