Rule-based Programming

Assignment #3

Excercise 1 (Adding Natural Numbers)**.** We denote natural numbers in successor notation, e.g. the number `3` is denoted as `s(s(s(0)))`. Write a program that adds natural numbers. Use a constraint `sum/1` for the resulting sum and `add/1` for each number to add. You may assume that one constraint `sum(0)` is present in the query.
E.g. the input `add(s(0)),add(s(s(0))), sum(0)` should produce the output `sum(s(s(s(0))))`.

Excercise 2 (Exchange Sort)**.** In the following, we represent an array as a multiset of pairs of the form `a(Index,Value)`. Implement exchange sort as follows and test it with an appropriate input:

```
a(I,V),a(J,W) <=> I>J, V<W | a(I,W),a(J,V).
```

What happens if the rule is run backwards, i.e. head and body of the rule are exchanged?
Assume we implement exchange sort in the following manner:

```
a(I,V),a(J,W) <=> I=:=J+1, V<W | a(I,W),a(J,V).
```

Can we still be sure that the final array will be sorted. Why?/Why not?

Excercise 3 (Newton)**.** Implement Newton's method for approximating square roots as follows:

```
:- use_module( library( chr )).
:- chr_constraint sqrt/1, sqrt/2.

% data format: sqrt( Argument, Approximation )

sqrt(X) <=> sqrt(X,1).
sqrt(X,G) <=> abs(G*G/X-1)>0.00001 | G1 is (G+X/G)/2, sqrt(X,G1).
```

Test the implementation with several example. What is the accuracy of the approximation?
Modify the program such that the approximation cycle is executed exactly 20 times, regardless of the accuracy of the approximation.

Excercise 4 (Reciprocal)**.** Implement the approximation of the reciprocal using the formula $G_{i+1} = 2G_i - XG_i^2$.