SEIII - Logikprogrammierung

Übungsblatt 08

Nico Hahn 6990715

Hieu Nguyen 6632126

Aufgabe 2

```
% Aufgabe 2.1
% binary prueft ob die Eingabe eine Binaerdarstellung oder nicht
is_even([]).
is_even([0|_]).
% Binaerdarstellung von einer geraden/ungeraden Zahl
even(X) :- binary(X), is_even(X).
odd(X) :- binary(X), not(is_even(X)).

% Aufgabe 2.2
% verdoppeln(+X,-Result)
verdoppeln(X, [0|X]) :- binary(X).
halbieren([_|X], X) :- binary(X).

% Aufgabe 2.3
% evenize(+UngeradeZahl, -GeradeZahl)
evenize([1|X], [0|X]) :- binary(X).

% Aufgabe 2.4
% Das Verbot führender Nullen ist berücksichtigt
% binary(+BinaryList)
binary([]).
binary([1|X]) :- binary(X).
binary([0|X]) :- binary(X), not(X = []).

% Aufgabe 2.5
% bin2dec(+Bin, -Dec)
bin2dec([], 0).
bin2dec([0|Z], X) :-
  bin2dec(Z, DEC),
  X is DEC * 2.
bin2dec([1|Z], X) :-
  bin2dec(Z, DEC),
  X is DEC * 2 + 1.

% dec2bin(+Dec, -Bin)
dec2bin(0, []).
dec2bin(X, [0|Z]) :-
 0 is X mod 2,
```

```
 NeuX is X div 2,
 dec2bin(NeuX, Z).
dec2bin(X, [1|Z]) :-
 1 is X mod 2,
 NeuX is X div 2,
 dec2bin(NeuX, Z).

% Aufgabe 2.6
% add(+ErstesBit, +ZweitesBit, +UebertragIn, Summe, UebertragOut)
add_bit(0, 0, 0, 0, 0).
add_bit(0, 1, 0, 1, 0).
add_bit(1, 0, 0, 1, 0).
add_bit(1, 1, 0, 0, 1).
add_bit(0, 0, 1, 1, 0).
add_bit(0, 1, 1, 0, 1).
add_bit(1, 0, 1, 0, 1).
add_bit(1, 1, 1, 1, 1).

% Aufgabe 2.7
% add_help(+Operand1,+Operand2,Uebertrag,Summe).
add(X,Y,S) :- add_help(X,Y,0,S).
add_help([], [], 1, [1]).
add_help([], [], 0, []).
add_help([X|Z], [], UI, [S|Rest]) :-
  add_bit(X,0,UI,S,_),
  add_help([],Z,0,Rest).
add_help([], [X|Z], UI, [S|Rest]) :-
  add_bit(X,0,UI,S,_),
  add_help([],Z,0,Rest).
add_help([X|Z1], [Y|Z2], UI, [S|Rest]) :-
  add_bit(X,Y,UI,S,UO),
  add_help(Z1,Z2,UO,Rest).

% Aufgabe 2.8
% mult(+Mult1, +Mult2, -Produkt)
mult(Y,X,P) :- mult_help(Y,X,[],P).
mult_help([1],X,Acc,Z) :- add(Acc,X,Z).
mult_help(Y,X,Acc,Z) :-
  even(Y),
  halbieren(Y, NeuY),
  verdoppeln(X, NeuX),
  mult_help(NeuY, NeuX, Acc, Z).
mult_help(Y,X,Acc,Z) :-
  odd(Y),
  add(Acc,X,NeuAcc),
  halbieren(Y, NeuY),
  verdoppeln(X, NeuX),
  mult_help(NeuY, NeuX, NeuAcc, Z).
```