Lösungsvorschlag

Aufgabe 19

Schreiben Sie ein Programm zur Berechnung der Potenzmenge einer Menge. Zur Kontrolle soll das Programm zunächst die Elemente der EingabeMenge zählen N und dann die Elemente der Ausgabemenge M. Beide Zahlen sollen auch ausgegeben werden.
In einem Goal soll geprüft werden ob $M = 2^N$ gilt.

```
%Aufgabe19
% Abbruchfall: die Potenzmenge der leeren Menge ist die Menge,die die leere Menge enthält
potenz_menge([],[[]]).
 %die Potenzmenge einer Menge mit N Elementen ergibt sich aus der Potenzmenge der Menge mit N-1 Elementen verkettet mit den zusätzlichen Teilmengen
potenz_menge([X|Xs],L):-potenz_menge(Xs,L1),
                        mischung(X,L1,L2),
                         append(L2,L1,L).

% das zusätzliche Element 'n' mischt sich mit den Potenzmengen 'n-1'.
% mischung(ZusätzlichesElement,BisherigePotenzmenge,ZusätzlicheMengen)
mischung(_,[],[]).
mischung(X,[Y|Ys],[Z|Zs]):-append([X],Y,Z),
                            mischung(X,Ys,Zs).

%Warum ist auch [X] drin?
%---------------------------------
```

Alternativlösung

```
%gleiche Idee mit Prädikat findall/3
potenzmenge([],[[]]).
potenzmenge([X|Y],L):- potenzmenge(Y,L1),
                       findall([X|Z],member(Z,L1),L2),
                       append(L1,L2,L).
```

```
potenzmenge_length:-
        write(user,'Geben Sie bitte eine Menge ein:\n'),
        read(user,X),
        potenzmenge(X,Y),
        length(X,N),
        length(Y,M),
        Z is 2**N,
        write('Die Anzahl der Elementen in der Menge '),
        write(X),
        write('ist N='),
        write(N),
        nl,
        write('Die Anzahl der Elementen in der Potenzmenge '),
        write(Y),
        write(' ist M='),
        write(M),
        write('=2**N='),
        write(2**N),
        write('='),
        write(Z).
```

Aufgabe 20

20.1)
Fügen Sie in das Tokenizerprogramm auf der Webseite eine Regel tokenize/4 ein, die führende Leerzeichen entfernt und diese wegwirft, es sollen also weder Token ' Haus' oder Token ' ' ausgegeben werden. Lesen Sie auch über den Zeilenumbruch hinweg. Zeilenumbrüche sollen wie ein Leerzeichen behandelt werden. Zeilenumbruch hat den Code 10.

20.2)
Fügen Sie Satzendeerkennung für Satzzeichen :,?,! ein.

20.3)
Versuchen Sie die Ordinalzahlenerkennung und die Abkürzungserkennung aus dem Leiss Tokenizer einzubauen.

```
%Aufgabe 20 Lösung für das Lesen von der Konsole
eingabe:-write(user_output,'Beeende die Eingabe mit <.><Return>'),
        nl,read_sentence(user_input,CharCodeList,[]),
        sentenceToAtoms(CharCodeList,AtomList),
        nl,writeq(AtomList).

/* Bis zum Satzende vom Strom lesen*/
read_sentence(Charlist,CharCode,SeenChars):-
        get_code(Charlist,Char),!,
        read_sentence(Charlist,CharCode,Char,SeenChars).


read_sentence(_Charlist,CharCode,Char,[C|RestSeenChars]):-
satzende([C,Char]),!,reverse([C|RestSeenChars],CharCode).

read_sentence(Charlist,CharCode,Char,SeenChars):-
        (   Char=32,
        SeenChars=[32|_])  % mehrere Leerzeichen ignorieren.
        -> read_sentence(Charlist,CharCode,SeenChars).

read_sentence(Charlist,CharCode,Char,SeenChars):-
        abkuerzung_erkennung(Char,SeenChars), %Abkuerzungerkennung
        !,read_sentence(Charlist,CharCode,[Char|SeenChars]).


read_sentence(Charlist,CharCode,Char,SeenChars):-
        (   member(Char,[33,44,58,59,63])  % Satzzeichen
        ->  read_sentence(Charlist,CharCode,[Char,32|SeenChars])).

read_sentence(Charlist,CharCode,Char,SeenChars):-
        ordinalzahlen([Char]),          % Ordinalzahlenerkennung
        !,read_sentence(Charlist,CharCode,[32,Char|SeenChars]).


read_sentence(Charlist,CharCode,Char,SeenChars):-
        read_sentence(Charlist,CharCode,[Char|SeenChars]).



sentenceToAtoms(CharCodeList,AtomList):- tokenize(CharCodeList,[],
[],AtomList).
```

```
tokenize([CharCode|RestCharCodes],[],AccAtomList,AtomList):-
  !,tokenize(RestCharCodes,[CharCode],AccAtomList,AtomList).

tokenize([CharCode|RestCharCodes],[],AccAtomList,AtomList):-
        (   leerzeichen(CharCode);zeilenende(CharCode)),
        !,tokenize(RestCharCodes,[],AccAtomList,AtomList).

tokenize([CharCode|RestCharCodes],Token,AccAtomList,AtomList):-
   (   zeilenende(CharCode);leerzeichen(CharCode)),!,
tokenToAtom(Token,Atom),
   tokenize(RestCharCodes,[],[Atom|AccAtomList],AtomList).


tokenize([],Token,AccAtomList,AtomList):-
   !, tokenToAtom(Token,Atom),reverse([Atom|AccAtomList],AtomList).

tokenize([],[],AccAtomList,AtomList):-
   !, reverse(AccAtomList,AtomList).

tokenize([CharCode|RestCharCodes],Token,AccAtomList,AtomList):-
   !, tokenize(RestCharCodes,[CharCode|Token],AccAtomList,AtomList).


tokenToAtom(RevToken,Atom):-
reverse(RevToken,Token),name(Atom,Token).

zeilenende(10).
satzende(".").
satzende(".\n").
leerzeichen(32).
punktzeichen(46).
abkuerzung("Dr").
abkuerzung("Prof").
ordinalzahlen([Zif]):-
        member(Zif,"0123456789").

abkuerzung_erkennung(Char,[C|Seen]):-
        leerzeichen(Char),
        punktzeichen(C),
        abkuerzung(X),
        reverse(X,Seen).
```

Aufgabe 21

21.1)
Öffnen Sie ein File und lesen Sie mit get_char(C) Zeichen ein bis das Dateiende erreicht ist. Geben Sie die Zeichen jeweils mit put_char aus. Immer wenn ein 'a' auftritt, sollen Sie stattdessen ein 'b' ausgeben.

```
/* Datei Lesen, danach Strom wieder zurück */
lese_datei(Name):-
open(Name,read,X,[encoding(utf8)]),
current_input(USER), set_input(X), lesen_und_schreiben,
set_input(USER),
close(X).

/*schreibfunktion*/

lesen_und_schreiben :- get_char(C), C \= end_of_file, schreibe(C),
lesen_und_schreiben.
lesen_und_schreiben.

schreibe(a) :-  put_char(b).
schreibe(C):-  put_char(C).
```

21.2)
Jetzt soll immer wenn nach einem 'a' ein 'b' auftritt stattdessen ein c ausgegeben werden.

```
/* Datei Lesen, danach Strom wieder zurück */
lese_datei(Name):-
open(Name,read,X,[encoding(utf8)]),
current_input(USER), set_input(X), lesen_und_schreiben([]),
set_input(USER),
close(X).

/*schreibfunktion*/

lesen_und_schreiben(X) :- get_char(C), C \= end_of_file, schreibe([C|X]),
lesen_und_schreiben(C).
lesen_und_schreiben(X).

schreibe([b|a]) :- put_char(c).
```

```
schreibe([C|_]):-  put_char(C).



%Aufgabe 21 Lösung für das Lesen von der Konsole
lesen_ein:-
        get_char(Stream),
        get_more(Stream).
get_more('.'):-!,fail.
get_more(a):-!,put_char(b),get_char(Next),get_more(Next).
get_more(Stream):-!,put_char(Stream),get_char(Next),((Stream =a,Next=b)->
get_more(c);get_more(Next)).


/*
21.1)
 ?- lesen_ein.
|: abcdefg.
bbcdefg
21.2)
?- lesen_ein.
|: agafaeacaaab.
agafaeacaaac
false.
?- lesen_ein.
|: abababab.
acacacac
false.

*/
```