```
%Aufgabe1.1
coins([200,100,50,20,10,5,2,1]).

%change(Coins, Change)
change(0,[]).
change(Coins,[C|RL]):-
        coins(C1),
        member(C,C1),
        C =< Coins,
        Coins1 is Coins - C,
        change(Coins1,RL).

/*
8 ?- change(5,X).
X = [5] ;
X = [2, 2, 1] ;
X = [2, 1, 2] ;
X = [2, 1, 1, 1] ;
X = [1, 2, 2] ;
X = [1, 2, 1, 1] ;
X = [1, 1, 2, 1] ;
X = [1, 1, 1, 2] ;
X = [1, 1, 1, 1, 1] ;

 ?- change(0,X).
X = [] .
*/

%Aufgabe1.2
change_sort(Coins,Change):-
change(Coins,Change),
sortiere(Change).

%sortiere(List)
sortiere([H| []]).
sortiere([H, F|RL]) :-
H @>= F,
sortiere([F|RL]).

/*
?- change_sort(5,X).
X = [5] ;
X = [2, 2, 1] ;
X = [2, 1, 1, 1] ;
X = [1, 1, 1, 1, 1] ;
*/
%Aufgabe1.3
change_min(Coins, Change):-
change_sort(Coins, Change),
!.
/*
9 ?- change_min(5,X).
X = [5].
*/

%Aufgabe1.4
/*

*/

%Aufgabe1.5

:- dynamic automat/1.
automat([200-3, 100-3, 50-3,  20-3, 10-3, 5-3, 2-3, 1-3]).

% get_coinnumber(+Coin, ?Number)
get_coinnumber(Coin, Number) :-
    automat(L),
    list_to_assoc(L, Assoc),
    get_assoc(Coin, Assoc, Number).
```

4

erst ausprobieren, dann begründen?
geht es auch effizienter?

2

2

```
%set_coinnumber(+Coin, ?Number)
set_coinnumber(Coin, Number):-
        automat(L),
        list_to_assoc(L,Assoc),
        get_assoc(Coin, Assoc, _, Assoc1, Number),
        assoc_to_list(Assoc1, L1),
        retractall(automat(_)),
        assert(automat(L1)).

%take_money(+Coinlist)
take_money([]).
take_money([H|RL]):-
        change_possible([H|RL]),
        get_coinnumber(H, T0),
        T1 is T0 - 1,
        set_coinnumber(H, T1),
        take_money(RL).

%add_money(+Coinlist)
add_money([]).
add_money([H|RL]):-
        get_coinnumber(H, T0),
        T1 is T0 + 1,
        set_coinnumber(H, T1),
        add_money(RL).

%count(+Member,+List,?Number)
count(_, [], 0).
count(X, [X | T], N) :-
  !, count(X, T, N1),
  N is N1 + 1.
count(X, [_ | T], N) :-
  count(X, T, N).

%change_possible(+Coinlist)
change_possible([]).
change_possible([H|RL]):-
        count(H,[H|RL], A),
        get_coinnumber(H,B),
        A =< B,
        change_possible(RL).

%possible_changes(+Coins, ?Change)
possible_changes(Coins, Change):-
        change(Coins, Change),
        sortiere(Change),
        change_possible(Change).

%pay(+Preis,+Coinlist,?Changes)
pay(Price, MoneyL, Changes):-
        sum_list(MoneyL,Money),
        Price =< Money,
        change_min(Price,Add),
        add_money(Add),
        Change is Money - Price,
        possible_changes(Change, Changes).
/*
?- automat(G).
G = [200-3, 100-3, 50-3, 20-3, 10-3, 5-3, 2-3, 1-3].

2 ?- add_money([2,5,10]).
true.

3 ?- automat(G).
G = [1-3, 2-4, 5-4, 10-4, 20-3, 50-3, 100-3, 200-3]. %%Geld hinzugefügt

4 ?- take_money([1,10]).
true .

5 ?- automat(G).
G = [1-2, 2-4, 5-4, 10-3, 20-3, 50-3, 100-3, 200-3]. %%Geld genommen
```

|| Das Wechselgeld kommt in die Kasse? Und die eingezahlten Beträge wandern in die Tasche des Verkäufers?

```
6 ?- take_money([1,1,1]).
false.

7 ?- automat(G).
G = [1-2, 2-4, 5-4, 10-3, 20-3, 50-3, 100-3, 200-3]. %%keine Änderungen

8 ?- possible_changes(5,X).
X = [5] ;
X = [2, 2, 1] ;
false.

9 ?- pay(45,[50],X).
X = [5] ;
X = [2, 2, 1] ;
false.

?- automat(G).
G = [1-2, 2-4, 5-5, 10-3, 20-5, 50-3, 100-3, 200-3]. %%es wurden 2x20 und 1x5 eingezahlt
                                                    wieso? Es wurde 1x50 eingezahlt?
?- pay(45,[5],X).
false.

?- automat(G).
G = [1-2, 2-4, 5-5, 10-3, 20-5, 50-3, 100-3, 200-3]. %%keine Änderungen
*/
```

3

11/14

```
%aus Skript Seite 125
trie2([[a, [b, [e, [n, [d, [*,abend]]]]],
[f, [f, [e, [*,affe]]]],
[l, [l, [e, [*,alle],
[s, [*,als]]]],
[s, [o, [*,also]]]]],
[b, [a, [u, [m, [*,baum]],
[e, [r, [*,aber]]]]]] ).

%Aufgabe 2.1
%Wir werden einen orthografischen Schlüssel verwenden, daher wird das 1. Argument als Schlüssel
benutzt.

%Aufgabe 2.2
% Fall 1 : Buchstabenbaum ist leer
trie([]).
% Fall 2 : Buchstabenbaum enthält Teilbäume
trie([H|T]) :-
    \+ \+ subtrie(H),
    trie(T).

% Teilbaum besteht aus Atom und Buchstabenbaum
subtrie([]).
subtrie([*|_]).
subtrie([H|T]):-
    atom(H),
    trie(T).

:- trie([]).
:- trie([[a,[*,info]]]).
:- trie([[a,[b,[*,info]],[c,[*,info]]]]).
:- \+ trie([a, [*,info]]).
:- \+ trie([a]).
:- \+ trie([a, [*,info]]).

%Aufgabe 2.3
%word2trie(+Liste, +Info, ?Baum)
word2trie(Liste, Info, [Baum]) :-
    word2trieInt(Liste, Info, Baum).

word2trieInt([], Info, [*,Info]).
word2trieInt([H|T], Info, [H,B2]) :-
    word2trieInt(T, Info, B2).

entry2trie(Entry, Trie) :-
    atom_chars(Entry, X),
    entry(Entry, Y),
    word2trie(X, Y, Trie).

%Aufgabe 2.4
%insert_entry(+Key, +Info, +OldT, ?NewT)

%Fall 1 : Liste mit Key ist bereits enthalten
%-Das erste Element (KeyH) des Keys wird extrahiert, der Rest des Keys ist KeyT
%-Die Liste mit KeyH als erstem Element wird aus dem alten Baum entfernt. Der Reduzierte alte Baum ist
RedT, der Rest der entfernten Liste DelT.
%-insert_entry wird mit KeyT und DelT rekursiv aufgerufen und liefert ZwT als neuen Baum. Dabei wird
DelT in eine Liste eingefügt, damit die Struktur erhalten bleibt
%-Ein neuer Teilbaum aus KeyH und ZwT wird erzeugt und RedT wieder angefügt. Da das Atom KeyH an
erster Stelle stehen muß, ist diese Reihenfolge wichtig.
insert_entry(Key, Info, OldT, [NewT]) :-
    Key = [KeyH|KeyT],
    select([KeyH, DelT], OldT, RedT),
    append([KeyH], ZwT, ZwT2),
    append(ZwT2, RedT, NewT),
%    write_ln('--- Fall 1 ---'),
%    format('~w~10|~w~n',['Key:',Key]),
%    format('~w~10|~w~n',['OldT:',OldT]),
%    format('~w~10|~w~n',['RedT:',RedT]),
%    format('~w~10|~w~n',['ZwT:',ZwT]),
%    format('~w~10|~w~n',['ZwT2:',ZwT2]),
%    format('~w~10|~w~n',['NewT:',NewT]),
```

```
    insert_entry(KeyT, Info, [DelT], ZwT).

%Fall 2 : Liste mit Key ist noch nicht enthalten
%-Es wird ein Buchstabenbaum aus dem Key und der Info aufgebaut und an den übergebenen Baum angefügt.
insert_entry(Key, Info, OldT, NewT) :-
    Key = [KeyH|_],
    \+ select([KeyH,_], OldT, _),
    word2trie(Key, Info, ZwT),
%    write_ln('  --- Fall 2 ---'),
%    format('  ~w~10|~w~n',['Key:',Key]),
%    format('  ~w~10|~w~n',['OldT:',OldT]),
%    format('  ~w~10|~w~n',['ZwT:',ZwT]),
%    format('  ~w~10|~w~n',['NewT:',NewT]),
    append(OldT, ZwT, NewT).

%Fall 3 : Schlüssel ist abgearbeitet
%-Die Information wird an den Baum OldT angefügt.
insert_entry([], Info, OldT, NewT) :-
%    write_ln('    --- Fall 3 ---'),
%    format('    ~w~10|~w~n',['OldT:',OldT]),
%    format('    ~w~10|~w~n',['Info:',Info]),
%    format('    ~w~10|~w~n',['NewT:',NewT]),
    append(OldT, [[*,Info]], NewT).

% Fall 1
:- insert_entry([a], info, [[a,[s,[*,info]]]],             [[a,[s,[*,info]],[*,info]]]).
:- insert_entry([a], info, [[a,[s,[*,info]]],[q,[*,info]]], [[a,[s,[*,info]],[*,info],[q,[*,info]]]]).

% Fall 2
:- insert_entry([a],   info, [],             [[a,[*,info]]]).
:- insert_entry([a],   info, [[l,[*,info]]], [[l,[*,info]],[a,[*,info]]]).
:- insert_entry([a,l], info, [[l,[*,info]]], [[l,[*,info]],[a,[l,[*,info]]]]).

% Fall 3
:- insert_entry([], info, [],                          [[*, info]]).
:- insert_entry([], info, [[l,[*,info]]],              [[l,[*,info]],[*,info]]).
:- insert_entry([], info, [[l,[*,info]],[s,[*,info]]], [[l,[*,info]],[s,[*,info]],[*,info]]).

%Aufgabe 2.5
%dict2trie(?Tree)
dict2trie(Tree) :-
    dictionary(Dict),
    dict2trie(Dict, [], Tree).

dict2trie([], T, T).
dict2trie([DictH|DictT], OldT, NewT) :-
    dict2trie(DictT, OldT, ZwT),
    DictH = entry(Key, Value),
    atom_chars(Key, KeyAtom),
    insert_entry(KeyAtom, Value, ZwT, NewT).
%    write_ln(NewT).

%Aufgabe 2.6
%Anmerkung: Die Tests funktionieren nur mit einem relativ kleinen Ausschnitt aus dem Wörterbuch,
ansonsten kommt es zu einem Stack-Überlauf.
% word2(?Word,+Trie,?Info).
word2([],[[*,Info]],Info).
word2([C|RW],[[C|RT]|_],Info) :- word2(RW,RT,Info).
word2(W,[_|Alt],Info) :- word2(W, Alt, Info).

/*
?- dict2trie(T),word2([z,w,a,r],T,I).
T = [[z, [w, [a, [*, [...|...]], [r|...]]], ['Z', [w, [e|...]], [z, [...|...]|...], [z|...]]]],
I = [ts, v, 'a:', r] .

?- dict2trie(T),word2(X,T,[ts, v, 'a:', r]).
T = [[z, [w, [a, [*, [...|...]], [r|...]]], ['Z', [w, [e|...]], [z, [...|...]|...], [z|...]]]],
X = [z, w, a, r] ;
*/
```

6

4

1

16 (26

27/30