Všechna řešení,

třídění, rozdílové seznamy

Všechna řešení

```
% z(Jmeno,Prijmeni,Pohlavi,Vek,Prace,Firma)
z(petr,novak,m,30,skladnik,skoda).  z(pavel,novy,m,40,mechanik,skoda).
z(rostislav,lucensky,m,50,technik,skoda).  z(alena,vesela,z,25,sekretarka,skoda).
z(jana,dankova,z,35,asistentka,skoda).  z(lenka,merinska,z,35,ucetni,skoda).
z(roman,maly,m,35,manazer,cs).  z(alena,novotna,z,40,ucitelka,zs_stara).
z(david,novy,m,30,ucitel,zs_stara).  z(petra,spickova,z,45,uklizecka,zs_stara).
```

- Najděte jméno a příjmení všech lidí.

```
?- findall(Jmeno-Prijmeni, z(Jmeno,Prijmeni,_,_,_,_),L).
?- bagof( Jmeno-Prijmeni, [S,V,Pr,F] ^ z(Jmeno,Prijmeni,S,V,Pr,F) ,  L).

?- bagof( Jmeno-Prijmeni, [V,Pr,F] ^ z(Jmeno,Prijmeni,S,V,Pr,F) ,  L ).
```

- Najděte jméno a příjmení všech zamestnanců firmy skoda a cs

```
?- findall( c(J,P,Firma), ( z(J,P,_,_,_,Firma), ( Firma=skoda ; Firma=cs ) ),
?- bagof( J-P, [S,V,Pr]^(z(J,P,S,V,Pr,F),( F=skoda ; F=cs ) ) ,  L ).
?- setof( P-J, [S,V,Pr]^(z(J,P,S,V,Pr,F),( F=skoda ; F=cs ) ) ,  L ).
```


Všechna řešení

Kolik žen a mužů je v databázi?

```
?- findall( c(P,J), z(P,J,z,_,_,_), L), length(L,N).
?- findall( c(P,J), z(P,J,m,_,_,_), L), length(L,N).

?- bagof(c(P,J), [Ve,Pr,Fi]^z(P,J,S,Ve,Pr,Fi), L), length(L,N).

?- findall( S-N, ( bagof(c(P,J), [Ve,Pr,Fi]^z(P,J,S,Ve,Pr,Fi), L),
                   length(L,N)
                 ), Dvojice ).
```


Všechna řešení: příklady

1. Jaká jsou příjmení všech žen?
2. Kteří lidé mají více než 30 roků? Nalezněte jejich jméno a příjmení.
3. Nalezněte abecedně seřazený seznam všech lidí.
4. Nalezněte příjmení vyučujících ze zs_stara.
5. Jsou v databázi dva bratři (mají stejné příjmení a různá jména)?
6. Které firmy v databázi mají více než jednoho zaměstnance?

```
1. findall(Prijmeni, z(_,Prijmeni,z,_,_,_), L).
2. findall(Jmeno-Prijmeni, ( z(Jmeno,Prijmeni,_,Vek,_,_), Vek>30 ), L).
3. setof(P-J, [S,V,Pr,F]^z(J,P,S,V,Pr,F),  L ).
4. findall(Prijmeni, ( z(_,Prijmeni,_,_,P,zs_stara), (P=ucitel;P=ucitelka) ), L).
5. findall(b(J1-P,J2-P), ( z(J1,P,m,_,_,_),z(J2,P,m,_,_,_), J1@<J2 ), L).
6. findall(F-Pocet, ( bagof(P, [J,S,V,Pr]^z(J,P,S,V,Pr,F), L),
                      length(L,Pocet), Pocet>1
                    ), S).
```

bubblesort(S,Sorted)

Seznam S seřaďte tak, že

- nalezněte první dva sousední prvky X a Y v S tak, že X>Y,
  vyměňte pořadí X a Y a získate S1; swap(S,S1)
  a seřaďte S1 rekurzivně bubblesortem
- pokud neexistuje žádný takový pár sousedních prvků X a Y,
  pak je S seřazený seznam

```
bubblesort(S,Sorted) :-
        swap (S,S1), !,            % Existuje použitelný swap v S?
        bubblesort(S1, Sorted).
bubblesort(Sorted,Sorted).         % Jinak je seznam seřazený

swap([X,Y|Rest],[Y,X|Rest1]) :-    % swap prvních dvou prvků
        X>Y.                       % nebo obecněji X@>Y, resp. gt(X,Y)
swap([X|Rest],[X|Rest1]) :-        % swap prvků až ve zbytku
        swap(Rest,Rest1).
```


quicksort(S,Sorted)

Neprázdný seznam S seřaďte tak, že konec rekurze pro S=[]

- vyberte nějaký prvek X z S; např. vyberte hlavu S
  rozdělte zbytek S na dva seznamy Small a Big tak, že:
  v Big jsou větší prvky než X a v Small jsou zbývající prvky split(X,Seznam,Small,Big)
- seřaďte Small do SortedSmall rekurzivně quicksortem
- seřaďte Big do SortedBig rekurzivně quicksortem
- setříděný seznam vznikne spojením SortedSmall a [X|SortedBig] append

```
quicksort([],[]).
quicksort([X|T], Sorted) :- split(X, Tail, Small, Big),
                            quicksort(Small, SortedSmall),
                            quicksort(Big, SortedBig),
                            append(SortedSmall, [X|SortedBig], Sorted).
split(X, [], [], []).
split(X, [Y|T], [Y|Small], Big) :- X>Y, !, split(X, T, Small, Big).
split(X, [Y|T], Small, [Y|Big]) :- split(X, T, Small, Big).
```


DÚ: insertsort(S,Sorted)

Neprázdný seznam S=[X|T] seřaďte tak, že konec rekurze pro S=[]

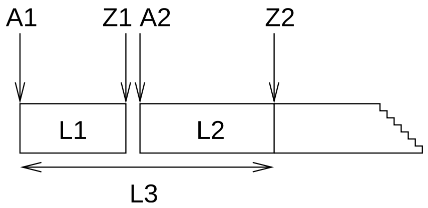
- seřaďte tělo T seznamu S rekurzivně insertsortem
- vložte hlavu X do seřazeného těla tak, insert(X,SortedT,Sorted)
  že výsledný seznam je zase seřazený.
  Víme: výsledek po vložení X je celý seřazený seznam.

```
insertsort([],[]).

insertsort([X|T],Sorted) :-
       insertsort(T,SortedT),          % seřazení těla
       insert(X,SortedT,Sorted).       % vložení X na vhodné místo

insert(X,[Y|Sorted],[Y|Sorted1]) :-
       X > Y, !,
       insert(X,Sorted,Sorted1).
insert(X,Sorted,[X|Sorted]).
```


Rozdílové seznamy

- Zapamatování konce a připojení na konec: rozdílové seznamy
- `[a,b] ... L1-L2 = [a,b|T]-T = [a,b,c|S]-[c|S] = [a,b,c]-[c]`
- Reprezentace prázdného seznamu: L-L
- 

- `?- append( [1,2,3|Z1]-Z1, [4,5|Z2]-Z2, A1-[]).`
- `append( A1-Z1, Z1-Z2, A1-Z2 ).`

```
          L1     L2     L3
append( [1,2,3,4,5]-[4,5], [4,5]-[], [1,2,3,4,5]-[] ).
```

`reverse(Seznam, Opacny)`

```
% kvadratická složitost
reverse( [], [] ).
reverse( [ H | T ], Opacny ) :-
        reverse( T, OpacnyT ),
        append( OpacnyT, [ H ], Opacny ).


% lineární složitost, rozdílové seznamy
reverse( Seznam, Opacny ) :- reverse0( Seznam, Opacny-[] ).
reverse0( [],  S-S ).
reverse0( [ H | T ], Opacny-OpacnyKonec  ) :-
         reverse0( T,  Opacny-[ H | OpacnyKonec]  ).
```


`quicksort` pomocí rozdílových seznamů

Neprázdný seznam S seřaďte tak, že

- vyberte nějaký prvek X z S;
  rozdělte zbytek S na dva seznamy Small a Big tak, že:
  v Big jsou větší prvky než X a v Small jsou zbývající prvky
- seřaďte Small do SortedSmall
- seřaďte Big do SortedBig
- setříděný seznam vznikne spojením SortedSmall a [X|SortedBig]

```
quicksort(S, Sorted) :- quicksort1(S,Sorted-[]).

quicksort1([],Z-Z).
quicksort1([X|T], A1-Z2) :-
        split(X, T, Small, Big),
        quicksort1(Small,  A1-[X|A2]),
        quicksort1(Big,  A2-Z2).
append(A1-A2, A2-Z2, A1-Z2).
```


DÚ: `palindrom(L)`

Napište predikát palindrom(Seznam), který uspěje pokud se Seznam čte stejně zezadu i zepředu, př. [a,b,c,b,a] nebo [12,15,1,1,15,12]

```
palindrom(Seznam) :- reverse(Seznam,Seznam).
```