

Seznamy, řez

Reprezentace seznamu

- **Seznam:** $[a, b, c]$, prázdný seznam $[]$
- **Hlava (libovolný objekt), tělo (seznam):** $.(Hlava, Te\l o)$

- všechny strukturované objekty stromy – i seznamy

- funktor ".", dva argumenty

- $.(a, .(b, .(c, []))) = [a, b, c]$

- notace: $[Hlava | Te\l o] = [a|Te\l o]$

Te\l o je v $[a|Te\l o]$ seznam, tedy píšeme $[a, b, c] = [a | [b, c]]$

- Lze psát i: $[a, b|Te\l o]$

- před "|" je libovolný počet prvků seznamu, za "|" je seznam zbývajících prvků

- $[a, b, c] = [a | [b, c]] = [a, b | [c]] = [a, b, c | []]$

- pozor: $[[a, b] | [c]] \neq [a, b | [c]]$

- Seznam jako **neúplná datová struktura:** $[a, b, c | T]$

- Seznam = $[a, b, c | T]$, $T = [d, e | S]$, Seznam = $[a, b, c, d, e | S]$

Seznamy a append

`append([], S, S).`

`append([X|S1], S2, [X|S3]) :- append(S1, S2, S3).`

Napište následující predikáty pomocí `append/3`:

● `prefix(S1, S2) :-`

DŮ: `suffix(S1,S2)`

Seznamy a append

`append([], S, S).`

`append([X|S1], S2, [X|S3]) :- append(S1, S2, S3).`

Napište následující predikáty pomocí `append/3`:

● `prefix(S1, S2) :- append(S1, _S3, S2).`

DŮ: `suffix(S1,S2)`

Seznamy a append

`append([], S, S).`

`append([X|S1], S2, [X|S3]) :- append(S1, S2, S3).`

Napište následující predikáty pomocí `append/3`:

● `prefix(S1, S2) :- append(S1, _S3, S2).`

DÚ: `suffix(S1,S2)`

● `last(X, S) :-`

`append([3,2], [6], [3,2,6]).` `X=6, S=[3,2,6]`

Seznamy a append

`append([], S, S).`

`append([X|S1], S2, [X|S3]) :- append(S1, S2, S3).`

Napište následující predikáty pomocí `append/3`:

● `prefix(S1, S2) :- append(S1, _S3, S2).`

DÚ: `suffix(S1,S2)`

● `last(X, S) :- append(_S1, [X], S).`

`append([3,2], [6], [3,2,6]).` `X=6, S=[3,2,6]`

Seznamy a append

`append([], S, S).`

`append([X|S1], S2, [X|S3]) :- append(S1, S2, S3).`

Napište následující predikáty pomocí `append/3`:

● `prefix(S1, S2) :- append(S1, _S3, S2).`

DÚ: `suffix(S1,S2)`

● `last(X, S) :- append(_S1, [X], S).`

`append([3,2], [6], [3,2,6]).` `X=6, S=[3,2,6]`

● `member(X, S) :-`

`append([3,4,1], [2,6], [3,4,1,2,6]).` `X=2, S=[3,4,1,2,6]`

DÚ: `adjacent(X,Y,S)`

Seznamy a append

`append([], S, S).`

`append([X|S1], S2, [X|S3]) :- append(S1, S2, S3).`

Napište následující predikáty pomocí `append/3`:

● `prefix(S1, S2) :- append(S1, _S3, S2).`

DÚ: `suffix(S1,S2)`

● `last(X, S) :- append(_S1, [X], S).`

`append([3,2], [6], [3,2,6]).` `X=6, S=[3,2,6]`

● `member(X, S) :- append(S1, [X|S2], S).`

`append([3,4,1], [2,6], [3,4,1,2,6]).` `X=2, S=[3,4,1,2,6]`

DÚ: `adjacent(X,Y,S)`

Seznamy a append

`append([], S, S).`

`append([X|S1], S2, [X|S3]) :- append(S1, S2, S3).`

Napište následující predikáty pomocí `append/3`:

● `prefix(S1, S2) :- append(S1, _S3, S2).`

DÚ: `suffix(S1,S2)`

● `last(X, S) :- append(_S1, [X], S).`

`append([3,2], [6], [3,2,6]).` `X=6, S=[3,2,6]`

● `member(X, S) :- append(S1, [X|S2], S).`

`append([3,4,1], [2,6], [3,4,1,2,6]).` `X=2, S=[3,4,1,2,6]`

DÚ: `adjacent(X,Y,S)`

● `% sublist(+S,+ASB)`

`sublist(S,ASB) :-`

Seznamy a append

`append([], S, S).`

`append([X|S1], S2, [X|S3]) :- append(S1, S2, S3).`

Napište následující predikáty pomocí `append/3`:

● `prefix(S1, S2) :- append(S1, _S3, S2).`

DÚ: `suffix(S1,S2)`

● `last(X, S) :- append(_S1, [X], S).`

`append([3,2], [6], [3,2,6]).` `X=6, S=[3,2,6]`

● `member(X, S) :- append(S1, [X|S2], S).`

`append([3,4,1], [2,6], [3,4,1,2,6]).` `X=2, S=[3,4,1,2,6]`

DÚ: `adjacent(X,Y,S)`

● `% sublist(+S,+ASB)`

`sublist(S,ASB) :- append(AS, B, ASB),`

`append(A, S, AS).`

POZOR na efektivitu, bez `append` lze často napsat efektivněji

Optimalizace posledního volání

● Last Call Optimization (LCO)

● Implementační technika snižující nároky na paměť

● Mnoho vnořených rekurzivních volání je náročné na paměť

● Použití LCO umožňuje vnořenou rekurzi s konstantními paměťovými nároky

● Typický příklad, kdy je možné použití LCO:

● procedura musí mít pouze jedno rekurzivní volání: **v posledním cíli poslední klauzule**

● cíle předcházející tomuto rekurzivnímu volání musí být **deterministické**

● `p(...) :- ...` % žádné rekurzivní volání v těle klauzule

`p(...) :- ...` % žádné rekurzivní volání v těle klauzule

...

`p(...) :- ..., !, p(...).` % řez zajišťuje determinismus

● Tento typ **rekurze lze převést na iteraci**

LCO a akumulátor

- Reformulace rekurzivní procedury, aby umožnila LCO
- Výpočet délky seznamu `length(Seznam, Deřka)`

`length([], 0)`.

`length([H | T], Deřka) :- length(T, Deřka0), Deřka is 1 + Deřka0.`

LCO a akumulátor

- Reformulace rekurzivní procedury, aby umožnila LCO
- Výpočet délky seznamu `length(Seznam, Délka)`

`length([], 0)`.

`length([H | T], Délka) :- length(T, Délka0), Délka is 1 + Délka0.`

- Upravená procedura, tak aby umožnila LCO:

```
% length( Seznam, ZapocitanaDélka, CelkovaDélka ):
```

```
%           CelkovaDélka = ZapocitanaDélka + „počet prvků v Seznam“
```

LCO a akumulátor

- Reformulace rekurzivní procedury, aby umožnila LCO
- Výpočet délky seznamu `length(Seznam, Delka)`

```
length( [], 0 ).
```

```
length( [ H | T ], Delka ) :- length( T, Delka0 ), Delka is 1 + Delka0.
```

- Upravená procedura, tak aby umožnila LCO:

```
% length( Seznam, ZapocitanaDelka, CelkovaDelka ):
```

```
%           CelkovaDelka = ZapocitanaDelka + „počet prvků v Seznam“
```

```
length( Seznam, Delka ) :- length( Seznam, 0, Delka ). % pomocný predikát
```

```
length( [], Delka, Delka ). % celková délka = započítaná délka
```

```
length( [ H | T ], A, Delka ) :- A0 is A + 1, length( T, A0, Delka ).
```

- Příkladový argument se nazývá **akumulátor**

Akumulátor a `sum_list(S, Sum)`

?- `sum_list([2,3,4], Sum)`.

bez akumulátoru:

Akumulátor a `sum_list(S, Sum)`

```
?- sum_list( [2,3,4], Sum ).
```

bez akumulátoru:

```
sum_list( [], 0 ).
```

```
sum_list( [H|T], Sum ) :- sum_list( T, SumT ),  
                          Sum is H + SumT.
```

s akumulátorem:

```
sum_list( S, Sum ) :- sum_list( S, 0, Sum ).
```

```
sum_list( [], Sum, Sum ).
```

```
sum_list( [H|T], A, Sum ) :- A1 is A + H,  
                             sum_list( T, A1, Sum).
```


Výpočet faktoriálu $\text{fact}(N, F)$

s akumulátorem:

Výpočet faktoriálu $\text{fact}(N, F)$

s akumulátorem:

```
fact( N, F ) :- fact ( N, 1, F ).
```

```
fact( 1, F, F ) :- !.
```

```
fact( N, A, F ) :- N > 1,  
                  A1 is N * A,  
                  N1 is N - 1,  
                  fact( N1, A1, F ).
```

```
r(X):-write(r1).
r(X):-p(X),write(r2).
r(X):-write(r3).

p(X):-write(p1).
p(X):-a(X),b(X),!,
      c(X),d(X),write(p2).
p(X):-write(p3).

a(X):-write(a1).
a(X):-write(a2).

b(X):- X > 0, write(b1).
b(X):- X < 0, write(b2).

c(X):- X mod 2 == 0, write(c1).
c(X):- X mod 3 == 0, write(c2).

d(X):- abs(X) < 10, write(d1).
d(X):- write(d2).
```

Prozkoumejte trasy výpočtu a navracení
např. pomocí následujících dotazů (vždy si
středníkem vyžádejte navracení):

- | | |
|------------------|--------------------|
| (1) $X=1, r(X).$ | (2) $X=3, r(X).$ |
| (3) $X=0, r(X).$ | (4) $X= -6, r(X).$ |

```

r(X):-write(r1).
r(X):-p(X),write(r2).
r(X):-write(r3).

p(X):-write(p1).
p(X):-a(X),b(X),!,
      c(X),d(X),write(p2).
p(X):-write(p3).

a(X):-write(a1).
a(X):-write(a2).

b(X):- X > 0, write(b1).
b(X):- X < 0, write(b2).

c(X):- X mod 2 == 0, write(c1).
c(X):- X mod 3 == 0, write(c2).

d(X):- abs(X) < 10, write(d1).
d(X):- write(d2).

```

Prozkoumejte trasy výpočtu a navracení např. pomocí následujících dotazů (vždy si středníkem vyžádejte navracení):

- (1) $X=1, r(X).$ (2) $X=3, r(X).$
(3) $X=0, r(X).$ (4) $X= -6, r(X).$

- řez v predikátu $p/1$ neovlivní alternativy predikátu $r/1$
- dokud nebyl proveden řez, alternativy predikátu $a/1$ se uplatňují, př. neúspěch $b/1$ v dotazu (3)
- při neúspěchu cíle za řezem se výpočet navrácí až k volající proceduře $r/1$, viz (1)
- alternativy vzniklé po provedení řezu se zachovávají - další možnosti predikátu $c/1$ viz (2) a (4)

r(X):-write(r1).

r(X):-p(X),write(r2).

r(X):-write(r3).

p(X):-write(p1).

p(X):-a(X),b(X),!,

 c(X),d(X),write(p2).

p(X):-write(p3).

a(X):-write(a1).

a(X):-write(a2).

b(X):- X > 0, write(b1).

b(X):- X < 0, write(b2).

c(X):- X mod 2 == 0, write(c1).

c(X):- X mod 3 == 0, write(c2).

d(X):- abs(X) < 10, write(d1).

d(X):- write(d2).

| ?- X=1,r(X).

r(X):-write(r1).

r(X):-p(X),write(r2).

r(X):-write(r3).

p(X):-write(p1).

p(X):-a(X),b(X),!,

 c(X),d(X),write(p2).

p(X):-write(p3).

a(X):-write(a1).

a(X):-write(a2).

b(X):- X > 0, write(b1).

b(X):- X < 0, write(b2).

c(X):- X mod 2 == 0, write(c1).

c(X):- X mod 3 == 0, write(c2).

d(X):- abs(X) < 10, write(d1).

d(X):- write(d2).

| ?- X=1,r(X).

r1

X = 1 ? ;

r(X):-write(r1).

r(X):-p(X),write(r2).

r(X):-write(r3).

p(X):-write(p1).

p(X):-a(X),b(X),!,

c(X),d(X),write(p2).

p(X):-write(p3).

a(X):-write(a1).

a(X):-write(a2).

b(X):- X > 0, write(b1).

b(X):- X < 0, write(b2).

c(X):- X mod 2 == 0, write(c1).

c(X):- X mod 3 == 0, write(c2).

d(X):- abs(X) < 10, write(d1).

d(X):- write(d2).

| ?- X=1,r(X).

r(X):-write(r1).

r1

r(X):-p(X),write(r2).

X = 1 ? ;

r(X):-write(r3).

p1r2

p(X):-write(p1).

X = 1 ? ;

p(X):-a(X),b(X),!,

c(X),d(X),write(p2).

p(X):-write(p3).

a(X):-write(a1).

a(X):-write(a2).

b(X):- X > 0, write(b1).

b(X):- X < 0, write(b2).

c(X):- X mod 2 == 0, write(c1).

c(X):- X mod 3 == 0, write(c2).

d(X):- abs(X) < 10, write(d1).

d(X):- write(d2).

r(X):-write(r1).	?- X=1,r(X).
r(X):-p(X),write(r2).	r1
r(X):-write(r3).	X = 1 ? ;
p(X):-write(p1).	p1r2
p(X):-a(X),b(X),!,	X = 1 ? ;
c(X),d(X),write(p2).	a1b1r3
p(X):-write(p3).	X = 1 ? ;
a(X):-write(a1).	
a(X):-write(a2).	
b(X):- X > 0, write(b1).	
b(X):- X < 0, write(b2).	
c(X):- X mod 2 == 0, write(c1).	
c(X):- X mod 3 == 0, write(c2).	
d(X):- abs(X) < 10, write(d1).	
d(X):- write(d2).	

r(X):-write(r1).	?- X=1,r(X).
r(X):-p(X),write(r2).	r1
r(X):-write(r3).	X = 1 ? ;
p(X):-write(p1).	p1r2
p(X):-a(X),b(X),!,	X = 1 ? ;
c(X),d(X),write(p2).	a1b1r3
p(X):-write(p3).	X = 1 ? ;
a(X):-write(a1).	no
a(X):-write(a2).	
b(X):- X > 0, write(b1).	
b(X):- X < 0, write(b2).	
c(X):- X mod 2 == 0, write(c1).	
c(X):- X mod 3 == 0, write(c2).	
d(X):- abs(X) < 10, write(d1).	
d(X):- write(d2).	

r(X):-write(r1).	?- X=1,r(X).
r(X):-p(X),write(r2).	r1
r(X):-write(r3).	X = 1 ? ;
	p1r2
p(X):-write(p1).	X = 1 ? ;
p(X):-a(X),b(X),!,	a1b1r3
c(X),d(X),write(p2).	X = 1 ? ;
p(X):-write(p3).	no
a(X):-write(a1).	?- X=0,r(X).
a(X):-write(a2).	
b(X):- X > 0, write(b1).	
b(X):- X < 0, write(b2).	
c(X):- X mod 2 ::= 0, write(c1).	
c(X):- X mod 3 ::= 0, write(c2).	
d(X):- abs(X) < 10, write(d1).	
d(X):- write(d2).	

r(X):-write(r1).	?- X=1,r(X).
r(X):-p(X),write(r2).	r1
r(X):-write(r3).	X = 1 ? ;
	p1r2
p(X):-write(p1).	X = 1 ? ;
p(X):-a(X),b(X),!,	a1b1r3
c(X),d(X),write(p2).	X = 1 ? ;
p(X):-write(p3).	no
a(X):-write(a1).	?- X=0,r(X).
a(X):-write(a2).	r1
b(X):- X > 0, write(b1).	X = 0 ? ;
b(X):- X < 0, write(b2).	
c(X):- X mod 2 ::= 0, write(c1).	
c(X):- X mod 3 ::= 0, write(c2).	
d(X):- abs(X) < 10, write(d1).	
d(X):- write(d2).	

r(X):-write(r1).	?- X=1,r(X).
r(X):-p(X),write(r2).	r1
r(X):-write(r3).	X = 1 ? ;
	p1r2
p(X):-write(p1).	X = 1 ? ;
p(X):-a(X),b(X),!,	a1b1r3
c(X),d(X),write(p2).	X = 1 ? ;
p(X):-write(p3).	no
a(X):-write(a1).	?- X=0,r(X).
a(X):-write(a2).	r1
b(X):- X > 0, write(b1).	X = 0 ? ;
b(X):- X < 0, write(b2).	p1r2
	X = 0 ? ;
c(X):- X mod 2 ::= 0, write(c1).	
c(X):- X mod 3 ::= 0, write(c2).	
d(X):- abs(X) < 10, write(d1).	
d(X):- write(d2).	

r(X):-write(r1).	?- X=1,r(X).
r(X):-p(X),write(r2).	r1
r(X):-write(r3).	X = 1 ? ;
	p1r2
p(X):-write(p1).	X = 1 ? ;
p(X):-a(X),b(X),!,	a1b1r3
c(X),d(X),write(p2).	X = 1 ? ;
p(X):-write(p3).	no
a(X):-write(a1).	?- X=0,r(X).
a(X):-write(a2).	r1
b(X):- X > 0, write(b1).	X = 0 ? ;
b(X):- X < 0, write(b2).	p1r2
	X = 0 ? ;
c(X):- X mod 2 == 0, write(c1).	a1a2p3r2
c(X):- X mod 3 == 0, write(c2).	X = 0 ? ;
d(X):- abs(X) < 10, write(d1).	
d(X):- write(d2).	

r(X):-write(r1).	?- X=1,r(X).
r(X):-p(X),write(r2).	r1
r(X):-write(r3).	X = 1 ? ;
	p1r2
p(X):-write(p1).	X = 1 ? ;
p(X):-a(X),b(X),!,	a1b1r3
c(X),d(X),write(p2).	X = 1 ? ;
p(X):-write(p3).	no
a(X):-write(a1).	?- X=0,r(X).
a(X):-write(a2).	r1
b(X):- X > 0, write(b1).	X = 0 ? ;
b(X):- X < 0, write(b2).	p1r2
	X = 0 ? ;
c(X):- X mod 2 == 0, write(c1).	a1a2p3r2
c(X):- X mod 3 == 0, write(c2).	X = 0 ? ;
d(X):- abs(X) < 10, write(d1).	r3
d(X):- write(d2).	X = 0 ? ;

r(X):-write(r1).	?- X=1,r(X).
r(X):-p(X),write(r2).	r1
r(X):-write(r3).	X = 1 ? ;
	p1r2
p(X):-write(p1).	X = 1 ? ;
p(X):-a(X),b(X),!,	a1b1r3
c(X),d(X),write(p2).	X = 1 ? ;
p(X):-write(p3).	no
a(X):-write(a1).	?- X=0,r(X).
a(X):-write(a2).	r1
b(X):- X > 0, write(b1).	X = 0 ? ;
b(X):- X < 0, write(b2).	p1r2
	X = 0 ? ;
c(X):- X mod 2 == 0, write(c1).	a1a2p3r2
c(X):- X mod 3 == 0, write(c2).	X = 0 ? ;
d(X):- abs(X) < 10, write(d1).	r3
d(X):- write(d2).	X = 0 ? ;
	no

r(X):-write(r1).	?- X=1,r(X).	
r(X):-p(X),write(r2).	r1	
r(X):-write(r3).	X = 1 ? ;	
p(X):-write(p1).	p1r2	?- X=3,r(X).
p(X):-a(X),b(X),!,	X = 1 ? ;	
c(X),d(X),write(p2).	a1b1r3	
p(X):-write(p3).	X = 1 ? ;	
	no	
a(X):-write(a1).	?- X=0,r(X).	
a(X):-write(a2).	r1	
b(X):- X > 0, write(b1).	X = 0 ? ;	
b(X):- X < 0, write(b2).	p1r2	
c(X):- X mod 2 == 0, write(c1).	X = 0 ? ;	
c(X):- X mod 3 == 0, write(c2).	a1a2p3r2	
d(X):- abs(X) < 10, write(d1).	X = 0 ? ;	
d(X):- write(d2).	r3	
	X = 0 ? ;	
	no	

r(X):-write(r1).

r(X):-p(X),write(r2).

r(X):-write(r3).

p(X):-write(p1).

p(X):-a(X),b(X),!,
c(X),d(X),write(p2).

p(X):-write(p3).

a(X):-write(a1).

a(X):-write(a2).

b(X):- X > 0, write(b1).

b(X):- X < 0, write(b2).

c(X):- X mod 2 == 0, write(c1).

c(X):- X mod 3 == 0, write(c2).

d(X):- abs(X) < 10, write(d1).

d(X):- write(d2).

| ?- X=1,r(X).

r1

X = 1 ? ;

p1r2

X = 1 ? ;

a1b1r3

X = 1 ? ;

no

| ?- X=0,r(X).

r1

X = 0 ? ;

p1r2

X = 0 ? ;

a1a2p3r2

X = 0 ? ;

r3

X = 0 ? ;

no

| ?- X=3,r(X).

r1

X = 3 ? ;

r(X):-write(r1).

r(X):-p(X),write(r2).

r(X):-write(r3).

p(X):-write(p1).

p(X):-a(X),b(X),!,
c(X),d(X),write(p2).

p(X):-write(p3).

a(X):-write(a1).

a(X):-write(a2).

b(X):- X > 0, write(b1).

b(X):- X < 0, write(b2).

c(X):- X mod 2 == 0, write(c1).

c(X):- X mod 3 == 0, write(c2).

d(X):- abs(X) < 10, write(d1).

d(X):- write(d2).

| ?- X=1,r(X).

r1

X = 1 ? ;

p1r2

X = 1 ? ;

a1b1r3

X = 1 ? ;

no

| ?- X=3,r(X).

r1

X = 3 ? ;

p1r2

X = 3 ? ;

| ?- X=0,r(X).

r1

X = 0 ? ;

p1r2

X = 0 ? ;

a1a2p3r2

X = 0 ? ;

r3

X = 0 ? ;

no

r(X):-write(r1).

r(X):-p(X),write(r2).

r(X):-write(r3).

p(X):-write(p1).

p(X):-a(X),b(X),!,
c(X),d(X),write(p2).

p(X):-write(p3).

a(X):-write(a1).

a(X):-write(a2).

b(X):- X > 0, write(b1).

b(X):- X < 0, write(b2).

c(X):- X mod 2 == 0, write(c1).

c(X):- X mod 3 == 0, write(c2).

d(X):- abs(X) < 10, write(d1).

d(X):- write(d2).

| ?- X=1,r(X).

r1

X = 1 ? ;

p1r2

X = 1 ? ;

a1b1r3

X = 1 ? ;

no

| ?- X=0,r(X).

r1

X = 0 ? ;

p1r2

X = 0 ? ;

a1a2p3r2

X = 0 ? ;

r3

X = 0 ? ;

no

| ?- X=3,r(X).

r1

X = 3 ? ;

p1r2

X = 3 ? ;

a1b1c2d1p2r2

X = 3 ? ;

r(X):-write(r1).

r(X):-p(X),write(r2).

r(X):-write(r3).

p(X):-write(p1).

p(X):-a(X),b(X),!,
c(X),d(X),write(p2).

p(X):-write(p3).

a(X):-write(a1).

a(X):-write(a2).

b(X):- X > 0, write(b1).

b(X):- X < 0, write(b2).

c(X):- X mod 2 == 0, write(c1).

c(X):- X mod 3 == 0, write(c2).

d(X):- abs(X) < 10, write(d1).

d(X):- write(d2).

| ?- X=1,r(X).

r1

X = 1 ? ;

p1r2

X = 1 ? ;

a1b1r3

X = 1 ? ;

no

| ?- X=0,r(X).

r1

X = 0 ? ;

p1r2

X = 0 ? ;

a1a2p3r2

X = 0 ? ;

r3

X = 0 ? ;

no

| ?- X=3,r(X).

r1

X = 3 ? ;

p1r2

X = 3 ? ;

a1b1c2d1p2r2

X = 3 ? ;

d2p2r2

X = 3 ? ;

r(X):-write(r1).

r(X):-p(X),write(r2).

r(X):-write(r3).

p(X):-write(p1).

p(X):-a(X),b(X),!,
c(X),d(X),write(p2).

p(X):-write(p3).

a(X):-write(a1).

a(X):-write(a2).

b(X):- X > 0, write(b1).

b(X):- X < 0, write(b2).

c(X):- X mod 2 == 0, write(c1).

c(X):- X mod 3 == 0, write(c2).

d(X):- abs(X) < 10, write(d1).

d(X):- write(d2).

| ?- X=1,r(X).

r1

X = 1 ? ;

p1r2

X = 1 ? ;

a1b1r3

X = 1 ? ;

no

| ?- X=0,r(X).

r1

X = 0 ? ;

p1r2

X = 0 ? ;

a1a2p3r2

X = 0 ? ;

r3

X = 0 ? ;

no

| ?- X=3,r(X).

r1

X = 3 ? ;

p1r2

X = 3 ? ;

a1b1c2d1p2r2

X = 3 ? ;

d2p2r2

X = 3 ? ;

r3

X = 3 ? ;

r(X):-write(r1).

r(X):-p(X),write(r2).

r(X):-write(r3).

p(X):-write(p1).

p(X):-a(X),b(X),!,
c(X),d(X),write(p2).

p(X):-write(p3).

a(X):-write(a1).

a(X):-write(a2).

b(X):- X > 0, write(b1).

b(X):- X < 0, write(b2).

c(X):- X mod 2 == 0, write(c1).

c(X):- X mod 3 == 0, write(c2).

d(X):- abs(X) < 10, write(d1).

d(X):- write(d2).

| ?- X=1,r(X).

r1

X = 1 ? ;

p1r2

X = 1 ? ;

a1b1r3

X = 1 ? ;

no

| ?- X=0,r(X).

r1

X = 0 ? ;

p1r2

X = 0 ? ;

a1a2p3r2

X = 0 ? ;

r3

X = 0 ? ;

no

| ?- X=3,r(X).

r1

X = 3 ? ;

p1r2

X = 3 ? ;

a1b1c2d1p2r2

X = 3 ? ;

d2p2r2

X = 3 ? ;

r3

X = 3 ? ;

no

r(X):-write(r1).

r(X):-p(X),write(r2).

r(X):-write(r3).

p(X):-write(p1).

p(X):-a(X),b(X),!,
c(X),d(X),write(p2).

p(X):-write(p3).

a(X):-write(a1).

a(X):-write(a2).

b(X):- X > 0, write(b1).

b(X):- X < 0, write(b2).

c(X):- X mod 2 == 0, write(c1).

c(X):- X mod 3 == 0, write(c2).

d(X):- abs(X) < 10, write(d1).

d(X):- write(d2).

| ?- X=1,r(X).

r1

X = 1 ? ;

p1r2

X = 1 ? ;

a1b1r3

X = 1 ? ;

no

| ?- X=0,r(X).

r1

X = 0 ? ;

p1r2

X = 0 ? ;

a1a2p3r2

X = 0 ? ;

r3

X = 0 ? ;

no

| ?- X= -6, r(X).

| ?- X=3,r(X).

r1

X = 3 ? ;

p1r2

X = 3 ? ;

a1b1c2d1p2r2

X = 3 ? ;

d2p2r2

X = 3 ? ;

r3

X = 3 ? ;

no

r(X):-write(r1).	?- X=1,r(X).	
r(X):-p(X),write(r2).	r1	?- X= -6, r(X).
r(X):-write(r3).	X = 1 ? ;	r1
p(X):-write(p1).	p1r2	?- X=3,r(X).
p(X):-a(X),b(X),!,	X = 1 ? ;	X = -6 ? ;
c(X),d(X),write(p2).	a1b1r3	r1
p(X):-write(p3).	X = 1 ? ;	X = 3 ? ;
a(X):-write(a1).	no	p1r2
a(X):-write(a2).	?- X=0,r(X).	X = 3 ? ;
b(X):- X > 0, write(b1).	r1	a1b1c2d1p2r2
b(X):- X < 0, write(b2).	X = 0 ? ;	X = 3 ? ;
c(X):- X mod 2 ::= 0, write(c1).	p1r2	d2p2r2
c(X):- X mod 3 ::= 0, write(c2).	X = 0 ? ;	X = 3 ? ;
d(X):- abs(X) < 10, write(d1).	a1a2p3r2	r3
d(X):- write(d2).	X = 0 ? ;	X = 3 ? ;
	r3	no
	X = 0 ? ;	
	no	

r(X):-write(r1).	?- X=1,r(X).	
r(X):-p(X),write(r2).	r1	?- X= -6, r(X).
r(X):-write(r3).	X = 1 ? ;	r1
p(X):-write(p1).	p1r2	?- X=3,r(X).
p(X):-a(X),b(X),!,	X = 1 ? ;	X = -6 ? ;
c(X),d(X),write(p2).	a1b1r3	p1r2
p(X):-write(p3).	X = 1 ? ;	X = -6 ? ;
a(X):-write(a1).	no	p1r2
a(X):-write(a2).	?- X=0,r(X).	X = 3 ? ;
b(X):- X > 0, write(b1).	r1	a1b1c2d1p2r2
b(X):- X < 0, write(b2).	X = 0 ? ;	X = 3 ? ;
c(X):- X mod 2 ::= 0, write(c1).	p1r2	d2p2r2
c(X):- X mod 3 ::= 0, write(c2).	X = 0 ? ;	X = 3 ? ;
d(X):- abs(X) < 10, write(d1).	a1a2p3r2	r3
d(X):- write(d2).	X = 0 ? ;	X = 3 ? ;
	no	no

r(X):-write(r1).	?- X=1,r(X).	
r(X):-p(X),write(r2).	r1	?- X= -6, r(X).
r(X):-write(r3).	X = 1 ? ;	r1
p(X):-write(p1).	p1r2	?- X=3,r(X).
p(X):-a(X),b(X),!,	X = 1 ? ;	X = -6 ? ;
c(X),d(X),write(p2).	a1b1r3	p1r2
p(X):-write(p3).	X = 1 ? ;	X = -6 ? ;
	no	a1b2c1d1p2r2
a(X):-write(a1).	?- X=0,r(X).	X = 3 ? ;
a(X):-write(a2).	r1	d2p2r2
b(X):- X > 0, write(b1).	X = 0 ? ;	X = 3 ? ;
b(X):- X < 0, write(b2).	p1r2	r3
c(X):- X mod 2 ::= 0, write(c1).	X = 0 ? ;	X = 3 ? ;
c(X):- X mod 3 ::= 0, write(c2).	a1a2p3r2	no
d(X):- abs(X) < 10, write(d1).	X = 0 ? ;	
d(X):- write(d2).	r3	
	X = 0 ? ;	
	no	

r(X):-write(r1).	?- X=1,r(X).	
r(X):-p(X),write(r2).	r1	?- X= -6, r(X).
r(X):-write(r3).	X = 1 ? ;	r1
p(X):-write(p1).	p1r2	?- X=3,r(X).
p(X):-a(X),b(X),!,	X = 1 ? ;	X = -6 ? ;
c(X),d(X),write(p2).	a1b1r3	p1r2
p(X):-write(p3).	X = 1 ? ;	X = -6 ? ;
	no	a1b2c1d1p2r2
a(X):-write(a1).	?- X=0,r(X).	X = 3 ? ;
a(X):-write(a2).	r1	a1b1c2d1p2r2
	X = 0 ? ;	d2p2r2
b(X):- X > 0, write(b1).	p1r2	X = 3 ? ;
b(X):- X < 0, write(b2).	X = 0 ? ;	r3
	a1a2p3r2	X = 3 ? ;
c(X):- X mod 2 ::= 0, write(c1).	X = 0 ? ;	no
c(X):- X mod 3 ::= 0, write(c2).	r3	
d(X):- abs(X) < 10, write(d1).	X = 0 ? ;	
d(X):- write(d2).	no	

r(X):-write(r1).	?- X=1,r(X).	
r(X):-p(X),write(r2).	r1	?- X= -6, r(X).
r(X):-write(r3).	X = 1 ? ;	r1
p(X):-write(p1).	p1r2	?- X=3,r(X).
p(X):-a(X),b(X),!,	X = 1 ? ;	X = -6 ? ;
c(X),d(X),write(p2).	a1b1r3	p1r2
p(X):-write(p3).	X = 1 ? ;	X = -6 ? ;
	no	a1b2c1d1p2r2
a(X):-write(a1).		X = 3 ? ;
a(X):-write(a2).	?- X=0,r(X).	a1b1c2d1p2r2
	r1	d2p2r2
b(X):- X > 0, write(b1).	X = 0 ? ;	X = 3 ? ;
b(X):- X < 0, write(b2).	p1r2	X = -6 ? ;
	X = 0 ? ;	r3
c(X):- X mod 2 ::= 0, write(c1).	a1a2p3r2	X = 3 ? ;
c(X):- X mod 3 ::= 0, write(c2).	X = 0 ? ;	no
d(X):- abs(X) < 10, write(d1).	r3	
d(X):- write(d2).	X = 0 ? ;	
	no	

r(X):-write(r1).	?- X=1,r(X).	
r(X):-p(X),write(r2).	r1	?- X= -6, r(X).
r(X):-write(r3).	X = 1 ? ;	r1
p(X):-write(p1).	p1r2	?- X=3,r(X).
p(X):-a(X),b(X),!,	X = 1 ? ;	X = -6 ? ;
c(X),d(X),write(p2).	a1b1r3	p1r2
p(X):-write(p3).	X = 1 ? ;	X = -6 ? ;
	no	a1b2c1d1p2r2
a(X):-write(a1).	?- X=0,r(X).	X = 3 ? ;
a(X):-write(a2).	r1	X = -6 ? ;
b(X):- X > 0, write(b1).	X = 0 ? ;	d2p2r2
b(X):- X < 0, write(b2).	p1r2	X = -6 ? ;
c(X):- X mod 2 ::= 0, write(c1).	X = 0 ? ;	r3
c(X):- X mod 3 ::= 0, write(c2).	a1a2p3r2	d2p2r2
d(X):- abs(X) < 10, write(d1).	X = 0 ? ;	X = -6 ? ;
d(X):- write(d2).	r3	
	X = 0 ? ;	
	no	

r(X):-write(r1).	?- X=1,r(X).	
r(X):-p(X),write(r2).	r1	?- X= -6, r(X).
r(X):-write(r3).	X = 1 ? ;	r1
p(X):-write(p1).	p1r2	?- X=3,r(X).
p(X):-a(X),b(X),!,	X = 1 ? ;	X = -6 ? ;
c(X),d(X),write(p2).	a1b1r3	p1r2
p(X):-write(p3).	X = 1 ? ;	X = -6 ? ;
	no	a1b2c1d1p2r2
a(X):-write(a1).	?- X=0,r(X).	X = 3 ? ;
a(X):-write(a2).	r1	X = -6 ? ;
b(X):- X > 0, write(b1).	X = 0 ? ;	d2p2r2
b(X):- X < 0, write(b2).	p1r2	X = -6 ? ;
c(X):- X mod 2 ::= 0, write(c1).	X = 0 ? ;	r3
c(X):- X mod 3 ::= 0, write(c2).	a1a2p3r2	X = 3 ? ;
	X = 0 ? ;	no
d(X):- abs(X) < 10, write(d1).	r3	r3
d(X):- write(d2).	X = 0 ? ;	X = -6 ? ;
	no	

r(X):-write(r1).	?- X=1,r(X).	
r(X):-p(X),write(r2).	r1	?- X= -6, r(X).
r(X):-write(r3).	X = 1 ? ;	r1
p(X):-write(p1).	p1r2	X = -6 ? ;
p(X):-a(X),b(X),!,	X = 1 ? ;	p1r2
c(X),d(X),write(p2).	a1b1r3	X = -6 ? ;
p(X):-write(p3).	X = 1 ? ;	a1b2c1d1p2r2
	no	X = -6 ? ;
a(X):-write(a1).	?- X=0,r(X).	a1b1c2d1p2r2
a(X):-write(a2).	r1	d2p2r2
b(X):- X > 0, write(b1).	X = 0 ? ;	X = -6 ? ;
b(X):- X < 0, write(b2).	p1r2	c2d1p2r2
c(X):- X mod 2 ::= 0, write(c1).	X = 0 ? ;	X = -6 ? ;
c(X):- X mod 3 ::= 0, write(c2).	a1a2p3r2	d2p2r2
d(X):- abs(X) < 10, write(d1).	X = 0 ? ;	X = -6 ? ;
d(X):- write(d2).	r3	r3
	X = 0 ? ;	X = -6 ? ;
	no	no

Řez: maximum

Je tato definice predikátu max/3 korektní?

$\text{max}(X, Y, X) : -X \geq Y, ! .$

$\text{max}(X, Y, Y) .$

Řez: maximum

Je tato definice predikátu max/3 korektní?

$\text{max}(X, Y, X) : -X \geq Y, ! .$

$\text{max}(X, Y, Y) .$

Není, následující dotaz uspěje: ?- $\text{max}(2, 1, 1) .$

Uved'te dvě možnosti opravy, se zachováním použití řezu a bez.

Řez: maximum

Je tato definice predikátu max/3 korektní?

$\text{max}(X, Y, X) : -X \geq Y, ! .$

$\text{max}(X, Y, Y) .$

Není, následující dotaz uspěje: ?- $\text{max}(2, 1, 1) .$

Uved'te dvě možnosti opravy, se zachováním použití řezu a bez.

$\text{max}(X, Y, X) : -X \geq Y .$

$\text{max}(X, Y, Z) : -X \geq Y, !, Z=X .$

$\text{max}(X, Y, Y) : -Y > X .$

$\text{max}(X, Y, Y) .$

Problém byl v definici, v první verzi se tvrdilo: $X=Z \wedge X \geq Y \Rightarrow Z=X$

správná definice je: $X \geq Y \Rightarrow Z=X$

Při použití řezu je třeba striktně oddělit vstupní podmínky od výstupních unifikací a výpočtu.

Řez: member

Jaký je rozdíl mezi následujícími definicemi predikátů member/2. Ve kterých odpovědích se budou lišit? Vyzkoušejte např. pomocí member(X, [1,2,3]).

```
mem1(H, [H|_]).
```

```
mem1(H, [_|T]) :- mem1(H,T).
```

```
mem2(H, [H|_]) :- !.
```

```
mem2(H, [_|T]) :- mem2(H,T).
```

```
mem3(H, [K|_]) :- H==K.
```

```
mem3(H, [K|T]) :- H\==K, mem3(H,T).
```

Řez: member

Jaký je rozdíl mezi následujícími definicemi predikátů member/2. Ve kterých odpovědích se budou lišit? Vyzkoušejte např. pomocí member(X, [1,2,3]).

```
mem1(H, [H|_]) .
```

```
mem1(H, [_|T]) :- mem1(H,T) .
```

```
mem2(H, [H|_]) :- ! .
```

```
mem2(H, [_|T]) :- mem2(H,T) .
```

```
mem3(H, [K|_]) :- H==K .
```

```
mem3(H, [K|T]) :- H\==K, mem3(H,T) .
```

- mem1/2 vyhledá všechny výskyty, při porovnávání hledaného prvku s prvky seznamu může dojít k vázání proměnných (může sloužit ke generování všech prvků seznamu)

Řez: member

Jaký je rozdíl mezi následujícími definicemi predikátů member/2. Ve kterých odpovědích se budou lišit? Vyzkoušejte např. pomocí member(X, [1,2,3]).

```
mem1(H, [H|_]) .
```

```
mem1(H, [_|T]) :- mem1(H,T) .
```

```
mem2(H, [H|_]) :- ! .
```

```
mem2(H, [_|T]) :- mem2(H,T) .
```

```
mem3(H, [K|_]) :- H==K .
```

```
mem3(H, [K|T]) :- H\==K, mem3(H,T) .
```

- mem1/2 vyhledá všechny výskyty, při porovnávání hledaného prvku s prvky seznamu může dojít k vázání proměnných (může sloužit ke generování všech prvků seznamu)
- mem2/2 najde jenom první výskyt, taky váže proměnné

Řez: member

Jaký je rozdíl mezi následujícími definicemi predikátů member/2. Ve kterých odpovědích se budou lišit? Vyzkoušejte např. pomocí member(X, [1,2,3]).

```
mem1(H, [H|_]) .
```

```
mem1(H, [_|T]) :- mem1(H,T) .
```

```
mem2(H, [H|_]) :- ! .
```

```
mem2(H, [_|T]) :- mem2(H,T) .
```

```
mem3(H, [K|_]) :- H==K .
```

```
mem3(H, [K|T]) :- H\==K, mem3(H,T) .
```

- mem1/2 vyhledá všechny výskyty, při porovnávání hledaného prvku s prvky seznamu může dojít k vázání proměnných (může sloužit ke generování všech prvků seznamu)
- mem2/2 najde jenom první výskyt, taky váže proměnné
- mem3/2 najde jenom první výskyt, proměnné neváže (hledá pouze identické prvky)

Dokážete napsat variantu, která hledá jenom identické prvky a přitom najde všechny výskyty?

Řez: member

Jaký je rozdíl mezi následujícími definicemi predikátů member/2. Ve kterých odpovědích se budou lišit? Vyzkoušejte např. pomocí member(X, [1,2,3]).

mem1(H, [H|_]) .

mem1(H, [_|T]) :- mem1(H,T) .

mem2(H, [H|_]) :- ! .

mem2(H, [_|T]) :- mem2(H,T) .

mem3(H, [K|_]) :- H==K .

mem3(H, [K|T]) :- H\==K, mem3(H,T) .

- mem1/2 vyhledá všechny výskyty, při porovnávání hledaného prvku s prvky seznamu může dojít k vázání proměnných (může sloužit ke generování všech prvků seznamu)
- mem2/2 najde jenom první výskyt, taky váže proměnné
- mem3/2 najde jenom první výskyt, proměnné neváže (hledá pouze identické prvky)

Dokážete napsat variantu, která hledá jenom identické prvky a přitom najde všechny výskyty?

mem4(H,[K|_]) :- H==K. mem4(H,[K|T]) :- mem4(H,T).

Řez: delete

`delete(X, [X|S], S).`

`delete(X, [Y|S], [Y|S1]) :- delete(X,S,S1).`

Napište predikát `delete(X,S,S1)`, který odstraní všechny výskyty `X` (pokud se `X` v `S` nevyskytuje, tak predikát uspěje).

Řez: delete

```
delete( X, [X|S], S ).
```

```
delete( X, [Y|S], [Y|S1] ) :- delete(X,S,S1).
```

Napište predikát `delete(X,S,S1)`, který odstraní všechny výskyty `X` (pokud se `X` v `S` nevyskytuje, tak predikát uspěje).

```
delete( _X, [], [] ).
```

```
delete( X, [X|S], S1 ) :- !, delete(X,S,S1).
```

```
delete( X, [Y|S], [Y|S1] ) :- delete(X,S,S1).
```

Seznamy: $\text{intersection}(A, B, C)$

DÚ: Napište predikát pro výpočet průniku dvou seznamů.

Nápověda: využijte predikát `member/2`

DÚ: Napište predikát pro výpočtu rozdílu dvou seznamů. Nápověda: využijte predikát `member/2`