1. kis házi feladat: CLP(MiniB) megvalósítása

CLP(MiniB) jellemzése

- **Tartomány:** logikai értékek (`1` és `0`, igaz és hamis)
- **Függvények** (egyben korlát-relációk):

`~ P`	`P` hamis (*negáció*).
`P * Q`	`P` és `Q` mindegyike igaz (*konjunkció*).
`P + Q`	`P` és `Q` legalább egyike igaz (*diszjunkció*).
`P # Q`	`P` és `Q` pontosan egyike igaz (*kizáró vagy*).
`P =\= Q`	Ugyanaz mint `P # Q`.
`P =:= Q`	Ugyanaz mint `~(P # Q)`.

- A fenti függvényjelek többsége szabványos beépített operátor (ezek prioritását nem célszerű módosítani), a `~` és `#` operátorokat – a CLP(B) könyvtárral megegyezően – az alábbi módon javasoljuk deklarálni:

```
:- op(300, fy, ~).
:- op(500, yfx, #).
```

1. kis házi feladat: CLP(MiniB) megvalósítása

A megvalósítandó eljárások

- `sat(`*Kif*`)`, ahol *Kif* változókból, a `0`, `1` konstansokból a fenti müveletekkel felépített logikai kifejezés. Jelentése: A *Kif* logikai kifejezés igaz. A `sat/1` eljárás ne hozzon létre választási pontot! A benne szereplő változók behelyettesítése esetén minél előbb ébredjen fel, és végezze el a megfelelő következtetéseket (lásd a példákat alább)!
- `count(`*Es*`, `*N*`)`, ahol *Es* egy (változó-)lista, *N* adott természetes szám. Jelentése: Az *Es* listában pontosan *N* olyan elem van, amelynek értéke 1.
- `labeling(`*Változók*`)`. Behelyettesíti a *Változók*at `0`, `1` értekekre. Visszalépés esetén felsorolja az összes lehetséges értéket.

1. kis házi feladat: egy kis segítség

Mikor érdemes a korlátokat felébreszteni?

- Ha egy változó (pl. `A`) behelyettesített: `nonvar(A)` ébresztési feltétel
- Ha van két változó (pl. `A` és `Res`) amelyek azonossága eldönthető: `?=(A,Res)` ébresztési feltétel

```
~(A, Res) :-
        when( (nonvar(A); nonvar(Res); ?=(A,Res)),
              not(A,Res)
            ).

not(A, Res) :-
        (   nonvar(A) -> Res is 1-A       % nonvar(A) ébresztés
        ;   nonvar(Res) -> A is 1-Res     % nonvar(Res) ébresztés
        ;   A == Res -> fail              % ?=(A,Res) ébresztés
        ).
```

- A kétargumentumú műveletekből generált korlátok esetén (pl. `#(A,B,Res)`), a `when` részben 3 `nonvar` és 3 `?=` feltétel szükséges.
- **Fontos:** A `count` eljárás esetében nem várjuk el az argumentumlistában levő változók azonosság-vizsgálatát (csak `nonvar` feltételek kellenek.)

1. kis házi feladat

Futási példák

```
| ?- sat(A*B =:= (~A)+B).
                          --->   <...felfüggesztett célok...> ? ; no
| ?- sat(A*B =:= (~A)+B), labeling([A,B]).
                          --->   A = 1, B = 0 ? ; A = 1, B = 1 ? ; no
| ?- sat((A+B)*C=\=A*C+B), sat(A*B).
                          --->   A = 1, B = 1, C = 0 ? ; no
| ?- sat(~A =:= A).       --->   no

| ?- count([A,A,B], 2). --->   <...felfüggesztett célok...> ? ; no
| ?- count([A,A,B], 2), labeling([A]).
                          --->   A = 1, B = 0 ? ; no
| ?- count([A,A,B,B], 3), labeling([A,B]).
                          --->   no
```

1. kis házi feladat, példák (folyt.)

```
| ?- trace, ~(A, A).
 1 1 Call: ~(A,A) ?
 2 2 Call: when((nonvar(A);nonvar(A);?=(A,A)),not(A,A))?
 3 3 Call: not(A,A) ?
 4 4 Call: nonvar(A) ?
 4 4 Fail: nonvar(A) ?
 5 4 Call: nonvar(A) ?
 5 4 Fail: nonvar(A) ?
 6 4 Call: A==A ?
 6 4 Exit: A==A ?
 3 3 Fail: not(A,A) ?
 2 2 Fail: when((nonvar(A);nonvar(A);?=(A,A)),not(A,A))?
 1 1 Fail: ~(A,A) ?
no
| ?- sat(A*A=:=B).

                                B = A ? ; no
| ?- sat(A#A=:=B).

                                B = 0 ? ; no
| ?- sat(A+B=:=C), A=B.

                                B = A, C = A ? ; no
```