Laborator 2

❖ **Update pentru notare:**

- Se va acorda o nota si la seminar. Ea va duce la acordarea de pana la un punct in plus la nota de laborator. Daca nota de laborator va fi 11, se va lua in considerare la formarea notei finale la acest curs. Mai exact, va ajuta la rotunjirea in sus a notelor finale (de exemplu, nota 9.3 va deveni 10).
- Nota de la seminar va fi acordata pe baza unor exercitii pe care le veti rezolva, cu totii, singuri, respectiv pe baza activitatii voastre la tabla.

❖ Copiati urmatorul program Prolog intr-un fisier cu extensia.pl;

```
culoare(rosu).
culoare(galben).
culoare(alb).
frunze(alterne).
frunze(mari).
frunze(nervuri_paralele).
petale(multepetale).
petale(petalegalbene).
petale(grupate_trompeta).
fl(trandafir, rosu, alterne, multepetale).
fl(floarea_soarelui, galben, mari, petalegalbene).
fl(crin, alb, nervuri_paralele, grupate_trompeta).

floare(A, X, Y, Z):-culoare(X), frunze(Y), petale(Z),
fl(A,X,Y,Z).
```

•Setati trace –ul (debug, https://sicstus.sics.se/sicstus/docs/3.7.1/html/sicstus_9.html):

```
?-trace.
```

•Urmariti cum se efectueaza procesul de backtracking in urma interogarii:

```
?-floare(crin, X,Y,Z).
```

•Puteti elimina trace–ul astfel:

```
?-notrace.
```

❖ Predicatul **cut (!)**

Daca Prologul gaseste predicatul cut intr-o regula, nu va mai efectua backtracking. Toate deciziile luate pana la cut vor ramane finale.

❖ Predicatul **fail (fail)**

Fail, intr-o conjunctie de scopuri (de obicei la sfarsit), forteaza intrarea in procesul de backtracking;

Copiati programul de mai jos intr-un fisier cu extensia pl si interogati-l asa cum este indicat. Observati efectul predicatului fail.

```
getX('X1').
getZ('Z1').
getX('X2').
getZ('Z2').
wr:-getX(X), getZ(Z), wrXZ(X, Z), fail.
wrXZ(X, Z):-write(X), write(' '), write(Z), write(' '), nl, fail.
```

Interogare:

```
?- wr
```

- nl determina programul sa sara la linie noua.
- write(X) afiseaza pe ecran valoarea cu care a fost instantiat X

❖ Ce se intampla daca eliminati, pe rand cate unul dintre predicatele fail prezente in program?

❖ Operatori in Prolog

•**= =** egalitate de termeni

```
1+2= = 2+1 no
```

- **=:=** egalitate de valori numerice a doua expresii aritmetice

```
1+2=:=2+1 yes
```

- **\= =** termeni diferiti. Functioneaza si pentru a verifica daca doua valori numerice sunt diferite. Daca vreti sa verificati daca o variabila instantiata este diferita de o variabila neinstantiata, trebuie sa folositi \==.
- **=\=** valori numerice diferite.
- **=** unificare
- **\=** negatia unificarii
- operatorul **is** trebuie folosit atunci cand avem de efectuat o operatie aritmetica.

```
X is 3+2. % X va lua valoarea 5
```

- Mai mic sau egal: =<
- Mai mare sau egal: >=
- Si: ,
- Sau: ;
- Negatie: \+

❖ Exercitii:

I. Cum ar putea fi implementata in Prolog functia $(x) = \begin{cases} 0, x \leq 3 \\ 2, x \in (3,6] \\ 4, x > 6 \end{cases}$?

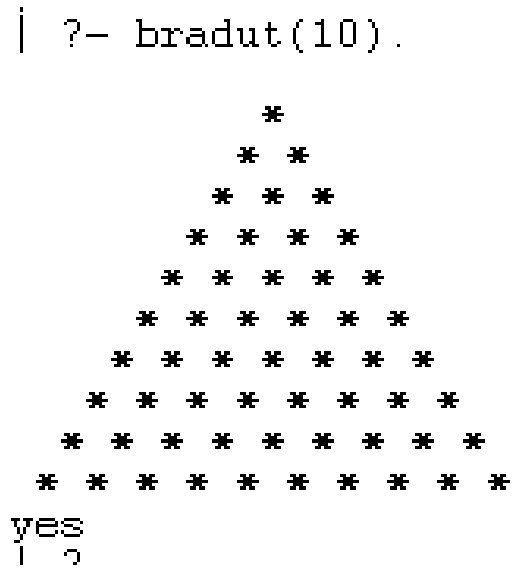
Intrati in modul debug si interogati in felul urmator: f(1,Y),2<Y. Este optima implementarea voastra? Daca nu, cum ati putea sa o optimizati?

II. Implementati un program care sa obtina factorialul unui numar dat.

III. Implementati un program Prolog care sa extraga cifrele din numere si sa afiseze pe ecran perechi, ca in urmatorul exemplu:

extrage(71) -> (7, sapte); (1, unu).

IV. Implementati un program prolog care sa deseneze un triunghi pe ecran, ca in imaginea de mai jos:



V. Modificati exercitiul anterior astfel incat sa desenati urmatorul bradut:

```
% [
| ?- bradut2(10).
                  *
                 *=*
                *===*
               *=====*
              *=======*
             *=========*
            *===========*
           *=============*
          *===============*
         *=================*
*************************
yes
```