# Laborator 3

## ➢Liste

• O lista este o secventa de oricate articole separate prin virgula.

[1,2,3,4,5,a,b,c]

• Lista vida se noteaza [].

• O lista nevida se poate imparti in cap si coada [A|B]. Capul A este un singur element. Coada B este lista. Putem pune in evidenta mai multe elemente la inceputul listei:

[A,B,C,…|T].

•Exemplu:

[1,2,3]=[1|[2,3]]=[1,2|[3]]=[1,2,3|[]].

•Putem colecta intr-o lista elemente ce satisfac o anumita proprietate cu ajutorul a 3 predicate predefinite:

1) **bagof(X,P,L)** pune in lista L elementele X ce satisfac P. Daca nu exista nici un astfel de element raspunsul este no.

2) **setof(X,P,L)** la fel ca bagof dar elimina duplicatele iar lista rezultata este sortata

3) **findall(X,P,L)** daca nu exista nici un element care sa satisfaca P rezultatul este yes iar L$\in \emptyset$ . Nu tine cont de variabilele care apar in P si nu apar in X.

Exemplu:

- Presupunand ca avem relatii de tipul baiat(nume,varsta), se se calculeze suma varstelor
baietilor din baza de cunostinte.

```
varstabaieti(L):-findall(Varsta,baiat(Nume,Varsta),L).
suma(S):-varstabaieti(L),suma(L,S).

suma([],0).
suma([H|T],S):-suma(T,S1), S is S1+H.
```

- Daca dorim sa obtinem toate solutiile pentru interogarea parinte(X,maria), folosim bagof sau setof:

```
parinte(ion,maria).
parinte(ana,maria).
parinte(ana,maria).
parinte(maria,elena).
parinte(maria,radu).
parinte(elena,nicu).
parinte(radu,gigi).
parinte(radu,dragos).
```

```
?- bagof(X ,parinte(X,maria), L).
L = [ion,ana,ana] ? ;                  % nu elimina duplicatele
no

?- setof(X, parinte(X,maria), L).
L = [ana,ion] ? ;                      % elimina duplicatele
no
```

•Adaugarea unui element in lista:

```
add_list([], L, L).
add_list(X, L, [X|L]).
```

•Concatenarea a doua liste:

```
lconcat([H|Tail], List2, [H|TailRez]):-lconcat(Tail, List2, TailRez).
lconcat([], L, L).
```

• Stergerea unui element din lista:

```
elimina(_,[],[]).
elimina(X,[X|Tail],T):-elimina(X,Tail,T),!.
elimina(X,[H|Tail],[H|T]):-elimina(X,Tail,T).
```

- ❑ Determinati numarul de elemente din lista;
- ❑ Verificati daca elementele unei liste se repeta (daca exista duplicate);
- ❑ Sa se elimine duplicatele dintr-o lista;
- ❑ Sa se insereze un element la inceputul listei, daca nu se regaseste deja in lista;
- ❑ Calculati reuniunea, intersectia si diferenta a doua multimi;
- ❑ Calculati suma elementelor pozitive dintr-o lista de intregi;

- Sa se inlocuiasca un element dintr-o lista (o singura aparitie si toate aparitiile);
- Sa se inverseze o lista;
- Sa se determine minimul unei liste;
- Sa se determine elementul de pe o anumita pozitie intr-o lista;
- Sa se insereze un element pe o anumita pozitie intr-o lista;
- Sa se interclaseze doua liste cu elemente intregi ordonate crescator;
- Sa se imparta o lista in doua subliste, in functie de o valoare data (in prima lista sa apara elementele mai mici iar in a doua lista sa apara elementele mai mari).

Sicstus Prolog are operatori predefiniti pentru liste si, pentru a-i putea folosi, trebuie incarcat pachetul:

?- use_module(library(lists)).

• ***append(?Prefix, ?Suffix, ?Combined).*** *Combined* este lista ce se formeaza prin concatenarea listelor *Prefix* si *Suffix*.

• ***delete(+List, +Element, ?Residue).*** *Residue* este lista rezultata in urma eliminarii ocurentelor lui *Element* in *List*.

• ***is_list(+List).*** *List* reprezinta o lista.

• ***last(?List, ?Last).*** *Last* este ultimul element din *List*.

• ***max_list(+ListOfNumbers, ?Max).*** *Max* este cel mai mare element din *ListOfNumbers*.

• ***member(?Element, ?List).*** *Element* este un element din *List*. Este folosit atat pentru a testa apartenenta unui element la o lista cat si pentru a enumera elementele din lista respectiva.

• ***min_list(+ListOfNumbers, ?Min).*** *Min* este cel mai mic element din *ListOfNumbers*.

• ***nextto(?X, ?Y, ?List).*** *X* si *Y* sunt elemente consecutive in *List*.

• ***no_doubles(?List).*** verifica daca *List contine sau* nu contine duplicate.

• ***non_member(?Element, ?List).*** *Element* nu este membru al *List*.

• ***nth(?N, ?List, ?Element).*** *Element* este elementul de pe pozitia N in *List*. Se considera ca primul element este pe pozitia 1.

• ***nth(?N, ?List, ?Element, ?Rest).*** *Element* este elementul de pe pozitia *N* in *List* iar *Rest* contine restul elementelor.

• ***remove_duplicates(+List, ?Pruned).*** *Pruned* este lista ce rezulta din eliminarea duplicatelor din *List*.

• ***reverse(?List, ?Reversed).*** *Reversed* este o lista ce contine elementele din List insa in ordinea inversa.

• ***substitute(+X, +Xlist, +Y, ?Ylist).*** *Ylist* este lista ce rezulta in urma inlocuirii elementelor X din Xlist cu Y.

***Mai multe informatii gasiti accesand urmatoarele link-uri:***
http://sicstus.sics.se/sicstus/docs/4.0.2/html/sicstus/lib_002dlists.html
http://sicstus.sics.se/sicstus/docs/3.7.1/html/sicstus_19.html