LISTE ÎN PROLOG

I. **Definirea listelor în PROLOG**

a. **Lista vida: []**

b. **Lista nevida cu valori simboluri: [10,20,30,40,50]**

c. **În Prolog – definire recursiva de forma: [H|T]**

[H|T] semnifică:

H- primul element din listă (Head), iar T- restul listei (Tail)

Exemplu: [1,2,3,4,5] => H=1, iar T=[2,3,4,5]

d. **Declararea domeniului lista în PROLOG:**

```
domains
        lista=tip_date*
```

Exemplu: lista= integer*, sau matrice=lista*

II. **Prelucrări asupra listelor**

1. **Verificarea apartenenței unui element într-o listă de numere reale. (X in L, unde L=[H1, H2, H3, …, Hn])**

Program PROLOG

```
domains
        lista=real*
predicates
        member(real, lista)
clauses
        member(X, [X|T]):-!.                    /* member(X, [X|_]):-!. */
        member(X, [H|T]):- member(X,T). /*member(X, [_|T]):- member(X,T). */
GOAL:   member(7, [3, 7, -2, 5, 9, 0])          member(-7, [3, 7, -2, 5, 9, 0])
```

2. **Lungimea (dimensiunea) unei liste date.**

Program PROLOG

```
domains
        lista=real*
predicates
        lungime(lista, integer)
```

```
clauses
        lungime([], 0).
        lungime([H|T], L):- lungime(T, L1), L=L1+1.
```

```
GOAL: lungime([3, 7, -2, 5, 9, 0], Lungime)
```

3. **Suma elementelor dintr-o listă de numere întregi.**

Program PROLOG

```
domains
        lista=integer*
predicates
        suma(lista, real)
clauses
        suma([], 0).
        suma([H|T], S):- suma(T, S1), S=S1+H.
```

```
GOAL: suma([3, 7, -2, 5, 9, 0], Suma)
```

4. **Concatenarea (alipirea) a două liste de numere reale (LR=L1L2). L1=[H1,H2,…,Hn], L2=[Z1,Z2,…,Zm], LR=[H1,H2,…,Hn, Z1,Z2,…,Zm] cu precizarea LR=[HR|TR], unde HR=H1, iar TR=[T1,L2], T1=restul listei L1.**

Program PROLOG

```
domains
        lista=integer*
predicates
        concatenare(lista, lista, lista)
        concatenare3(lista,lista,lista,lista)
clauses
        concatenare([], L2, L2).
        concatenare([H1|T1], L2, [H1|TR]):- concatenare(T1, L2, TR).
        concatenare3(L1, L2, L3, R):- concatenare(L1,L2,R1),
concatenare(R1,L3,R).
```

```
GOAL: concatenare([3,7,-2,5,9,0], [3,4,5,6,7], Rez)
        Concatenare3([3,7,-2,5,9,0], [3,4,5,6,7], [3,2,1,0,6], R)
```

5. **Numărul de apariții al unui element într-o listă de numere întregi.**

Program PROLOG

```
domains
        lista=integer*
```

```
predicates
        count(integer, lista, integer)
clauses
        count(X, [], 0).
        count(X, [X|T], R):- count(X, T, R1), R=R1+1, !.
        count(X, [H|T], R):- count(X, T, R).
```

GOAL: count(7, [3,7,-2,7,7,0,7,7], Rez)

6. **Ștergerea unui element dintr-o listă de numere reale.**
   a. **ștergerea primei apariții în lista dată**

```
sterge1(X, [X|T], T):-!.
sterge1(X, [Y|T], [Y|T1]):- sterge1(X, T, T1).
```

   b. **ștergerea tuturor aparițiilor în lista dată**

```
sterge2(X, [], []).
sterge2(X, [X|T], R):- sterge2(X,T,R), !.
sterge2(X, [Y|T], [Y|TR]):- sterge2(X, T, TR).
```

GOAL: sterge1(7, [3,7,-2,7,7,0,7,7], Rez) | sterge2(7, [3,7,-2,7,7,0,7,7], Rez)

7. **Împărțirea unei liste de numere reale în două liste, o listă cu valori pozitive, iar cealaltă cu valori negative.**

Program PROLOG

```
domains
        lista=real*
predicates
        impartire(lista, lista, lista)
clauses
  impartire([], [], []).
  impartire([H|T], [H|TPoz], LNeg):- H>=0, impartire(T, TPoz, LNeg), !.
  impartire([H|T], LPoz, [H|TNeg]):- impartire(T, LPoz, TNeg).
```

GOAL: impartire([3,7,-2,7,-7,0,7,-8], LPoz, LNeg)

III. Aplicații folosind listele

1. **Împărțirea unei liste de numere reale în două liste în funcție de un parametru real constant k.**

Program PROLOG

```
domains
        lista=integer*
predicates
        impartirek(integer, lista, lista, lista)
clauses
  impartirek(K, [], [], []).
  impartirek(K, [H|T], [H|TMari], LMici):- H>=K, impartirek(K, T, TMari, LMici), !.
  impartirek(K, [H|T], LMari, [H|TMici]):- impartirek(K, T, LMari, TMici).
```

GOAL: impartirek(7, [3,17,-2,7,-7,0,9,-8], LKmari, LKmici)

2. **Media aritmetică a valorilor dintr-o listă de numere întregi.**

```
/* media_aritmetica(lista, real)        */
/* media_aritmetica(L, Ma):- suma(L, S), lungime(L, N), Ma= S/N.  */
```

3. **Să se determine reversa (inversa) unei liste de numere întregi.**
   (Observație: Reversa([3 5 7 9])= [9, 7, 5, 3])

Program PROLOG

```
domains
        lista=integer*
predicates
        reversa(lista, lista)
        concatenare(lista,lista,lista)
clauses
        concatenare([], L2, L2).
        concatenare([H1|T1], L2, [H1|TR]):- concatenare(T1, L2, TR).
        reversa([], []).
        reversa([H|T], R):- reversa(T, R1), concatenare(R1,[H],R).
```

GOAL: reversa([3, 7, -2, 5, 9, 0], Rev)

4. **Să se calculeze suma numerelor pare dintr-o listă de numere întregi.**

Program PROLOG

```
domains
        lista=integer*
predicates
        sumapar(lista, integer)
```

```
        sumaimpar(lista, integer)
    clauses
        sumapar([],0).
        sumapar([H|T], Spar):- H mod 2 =0, sumapar(T,R), Spar=R+H.
        sumapar([H|T], Spar):- H mod 2 =1, sumapar(T, Spar).
        sumaimpar([],0).
        sumaimpar([H|T], Simpar):- H mod 2 =1, sumaimpar(T,R),
Simpar=R+H.
        sumaimpar([H|T], Simpar):-  H mod 2 =0, sumaimpar(T,Simpar).
```

GOAL: sumapar([3, 7, -2, 5, 9, 0], Spar) | sumaimpar([3, 7, -2, 5, 9], Simpar)

Suma numerelor pare și impare dintr-o listă de numere întregi.

```
sumaparimpar([],0,0).
sumaparimpar([H|T], Spar,Simpar):- H mod 2=0,
sumaparimpar(T,R1,Simpar), Spar=R1+H.
sumaparimpar([H|T], Spar,Simpar):- H mod 2 =1,
sumaparimpar(T,Spar,R2), Simpar=R2+H.
```

GOAL: sumaparimpar([3, 7, -2, 6, 19, 0], Spar, Simpar)

5. **Concatenarea a 3 liste de numere reale.**

```
/* concatenare3(lista, lista, lista, lista)    */
/* concatenare3(L1,L2,L3,R):- concatenare(L1, L2, R1), concatenare(R1,
L3, R).        */
```

6. **Să se determine intersecția a două mulțimi reprezentate prin liste.**

```
/* intersectie(lista, lista, lista)       */
/* intersectie(L1,L2,R):- ?.        */
```

7. **Să se determine reuniunea a două mulțimi reprezentate prin liste.**

```
/* reuniune(lista, lista, lista)        */
/* reuniune(L1,L2,R):- ?.        */
```

Program PROLOG

```
    domains
        lista=integer*
    predicates
        member(integer, lista)
        reuniune(lista, lista, lista)
    clauses
```

```
member(X, [X|T]):-!.
member(X, [_|T]):- member(X, T).
reuniune([], L2, L2).
reuniune([H|T], L2, [H|TR]):- not(member(H, L2)), reuniune(T, L2, TR), !.
reuniune([H|T], L2, R):- reuniune(T, L2, R).
```

GOAL: reuniune([3, 7, -2, 5], [5, 7, 2], Reuniune)

8. **Să se determine maximul elementelor dintr-o listă de numere reale.**

Program PROLOG

```
domains
        lista=integer*
predicates
        maxim(integer, integer, integer)
        maximL(lista, integer)
clauses
        maxim(A, B, A):-A>=B,!.
        maxim (A, B, B).

        maximL([X], X).
        maximL([H|T], Max):-maximL(T, MaxT), maxim(H, MaxT, Max).
```

GOAL: maximL([3, 7, -2, 5, 15, 7, 2], MaxLista)

9. **Să se determine maximul elementelor dintr-o matrice de numere reale.**

Program PROLOG

```
domains
        lista= integer*
        matrice= lista*
predicates
        maxim(integer, integer, integer)
        maximL(lista, integer)
        maximM(matrice, integer)
clauses
        maxim(A, B, A):-A>=B,!.
        maxim (A, B, B).

        maximL([X], X).
        maximL([H|T], Max):-maximL(T, MaxT), maxim(H, MaxT, Max).
```

```
maximM([X], M):- maximL(X, M).
maximM([H|T], MaxM):-maximM(T, MaxT1),  maximL(H, MaxT2), maxim(MaxT1, MaxT2, MaxM).
```

GOAL: maximM([[3, 7, -2], [5, 15, 7], [2, 3, -2]], MaxMatrix)

10. Să se calculeze produsul scalar a doi vectori de numere întregi.

```
/* produs_scalar(lista, lista, real) */
/* produs_scalar([],[],0).        */
/* produs_scalar([H1|T1],[H2|T2], R):- produs_scalar(T1,T2,R1), R=R1+H1*H2.     */
```

GOAL: produs_scalar([3, 7, -2], [5, 1, 2], Produs_scalar)

11. Înmulțirea unui vector cu o matrice.

```
/* inmultireVM(lista, matrice, lista)        */
/* inmultireVM(V,[],[]).         */
/* inmultireVM(V,[Hm|Tm], [Hr|Tr]):- produs_scalar(V,Hm,Hr), inmultireVM(V, Tm, Tr).     */
```

GOAL: inmultireVM([1, 0, -1], [[5, 1, 2], [1, 2, 0], [3, 1, 1]], Result_vector)

inmultireMM([[1, 0, -1],[2 1 1]], [[5, 1, 2], [1, 2, 0], [3, 1, 1]], Matrix_result)

12. Să se determine produsul a două matrici.

```
domains
        lista= integer*
        matrice= lista*
predicates
        produs_matrice(matrice,matrice,matrice)
    clauses
produs_matrice([],M,[]).
produs_matrice([H1|T1],M2,[X|R]):-inmultireVM(H1,M2,X), produs_matrice(T1,M2,R).
```

13. Să se determine concatenarea a N liste de numere întregi.

```
domains
        lista= integer*
        matrice= lista*
predicates
        concatenare(lista,lista,lista)
        concatenareN(matrice,lista)
  clauses
        concatenare([],L2,L2).
        concatenare([H|T],L2,[H|Tr]):-concatenare(T,L2,Tr).
        concatenareN([],[]).
        concatenareN([Hm|Tm],R):-concatenareN(Tm,Rm),
concatenare(Hm,Rm,R).
```

GOAL: concatenareN([[3, 7, -2], [5, 15], [7, 2], [9, -3, 7], [7, 8, -7]], R)

14.Să se determine lista maximelor din N liste de numere întregi.

//////////

GOAL: maxim_listN([[3, 7, -2], [5, 15], [7, 2], [9, -3, 7], [7, 8, -7]], R)

R=[7, 15, 7, 9, 8]

Tema:

1. **Sa se calculeze media aritmetica a valorilor din lista de maxime.**
2. **Sa se calculeze suma valorilor pare / impare din lista de maxime.**
3. **Sa se calculeze media aritmetica a valorilor obtinute prin concatenarea a N liste.**
4. **Sa se calculeze numarul de valori pare / impare din lista obtinuta prin concatenarea a N liste.**