1. Hàm member

```
member1(X,[X|T]).
member1(X,[Y|T]):-member1(X,T).
?-
|    member1(1,[1,2,3,4]).
Yes
?- member1(X,[1,1,2,3,4]).
X = 1 ;
X = 1 ;
X = 2 ;
X = 3 ;
X = 4 ;
No
?-
```

2.Hàm kiểm tra X có thuộc L hay không đây cũng chính là hàm member

```
kinL(X,[X|T]).
kinL(X,[Y|T]):-kinL(X,T).
?-
|    kinL(2,[1,2,3,4]).

Yes
?- kinL(1,[2,3,4]).

No
?- kinl(X,[1,2,3,5]).
Correct to: kinL(X, [1, 2, 3, 5])?
Please answer 'y' or 'n'? yes

X = 1 ;

X = 2 ;

X = 3 ;

X = 5 ;

No
?-
```

3.Hàm mylength để tính độ dài của một danh sách

Chú ý :Tên của một hàm phải bắt đầu bằng chữ cái thường

```
mylength([],0).
mylength([H|T],N):-mylength(T,M),N is M +1.
?-
|    mylength([],N).

N = 0 ;

No
?- mylength([1,2,3],N).

N = 3
```

Yes
?- mylength([],N).

N = 0

Yes
?- mylength([1,2,3],3).

Yes
?-
4.Hàm myappend(giống hàm concat) dùng để ghép hai danh sách thành một danh sách.
```
myappend([],L,L).
myappend([H|T],L,[H|L1]):-myappend(T,L,L1).
?-
|    myappend([1,2],[3,4],[1,2,3,4]).
```

Yes
?- myappend([1,2],L,[1,2,5,7]).

L = [5, 7]

Yes
?- myappend(H,[1,2],[1,2,3,4,1,2]).

H = [1, 2, 3, 4]

Yes
?-
5.Ghép X vào vị trí đầu tiên của danh sách L
```
ghepXinfirL(X,[],[X]).
ghepXinfirL(X,L,[X|L]).
?- ghepXinfirL(X,[4,3,5],[7,4,3,4]).
```

No
?- ghepXinfirL(X,L,[1,2,3,46]).

X = 1
L = [2, 3, 46]

Yes
?- ghepXinfirL(X,[2,3,7],L).
X = _G291
L = [_G291, 2, 3, 7]

Yes
?- ghepXinfirL(X,L,T).

X = _G246
L = []
T = [_G246] ;

X = _G246
L = _G247
T = [_G246|_G247] ;

No
?- ghepXinfirL(X,[3,6,7],[9,3,6,7]).

X = 9

Yes
?- ghepXinfirL(2,[3,4,1],[2,3,4,1]).

Yes
?-
6.Xoá phần tử X khỏi danh sách L
myremove(X,[X|L],L).
myremove(X,[H|L],[H|T]):-myremove(X,L,T).
?-
| myremove(X,[2,3,4],[]).

No
?- myremove(X,[2,3,4],[3,4]).

X = 2

Yes
?- myremove(X,[2,3,4],[2,4]).

X = 3 ;

No
?- myremove(X,[2,3,4],[2,3]).

X = 4
Yes
?- myremove(2,[2,4,5,2,6,2],L).

L = [4, 5, 2, 6, 2] ;

L = [2, 4, 5, 6, 2] ;

L = [2, 4, 5, 2, 6] ;

No
7.Chèn X vào vị trí bất kỳ trong danh sách L
myremove(X,[X|L],L).
myremove(X,[H|L],[H|T]):-myremove(X,L,T).
insertXinL(X,L,T):-myremove(X,T,L).
?-
| insertXinL(2,[3,4,5],L).

L = [2, 3, 4, 5] ;

L = [3, 2, 4, 5] ;

L = [3, 4, 2, 5] ;

L = [3, 4, 5, 2] ;

No
?- insertXinL(X,[2,1,5],[2,7,1,5]).

X = 7 ;

No
?- insertXinL(X,L,[3,5,2,8]).

X = 3
L = [5, 2, 8] ;

X = 5
L = [3, 2, 8] ;

X = 2
L = [3, 5, 8] ;

X = 8
L = [3, 5, 2] ;

No
?- insertXinL(X,[6,4,8,2],L).

X = _G300
L = [_G300, 6, 4, 8, 2] ;

X = _G300
L = [6, _G300, 4, 8, 2] ;

X = _G300
L = [6, 4, _G300, 8, 2] ;

X = _G300
L = [6, 4, 8, _G300, 2] ;
X = _G300
L = [6, 4, 8, 2, _G300] ;

No
8.Hàm reverse đảo ngược danh sách L các phần tử theo vị trí từ sau đến trước
reverse([],[]).
reverse([H|T],L):-reverse(T,L1),concat(L1,[H],L).
?-

| reverse([1,2,3],L).

L = [3, 2, 1]

Yes
?- reverse(L,[2,3,4]).

L = [4, 3, 2]

Yes
9. Hàm palindrone kiểm tra xem danh sách có đôi xứng không?

```
reverse([],[]).
reverse([H|T],L):-reverse(T,L1),concat(L1,[H],L).
palindrone(L):-reverse(L,L).
?-
| palindrone([1,2,3,2,1]).
```

Yes
?- palindrone([1,2,3,4]).

No
10.Hàm replace thay thế những phần tử có giá trị là x thành giá trị là k trong danh sách L

```
replace(X,K,[],[]).
replace(X,K,[X|L],[K|L1]):-replace(X,K,L,L1).
replace(X,K,[H|L],[H|L1]):-replace(X,K,L,L1).
?- replace(2,3,[],L).
```

L = []

Yes
?- replace(2,3,[1,3,4,5],L).

L = [1, 3, 4, 5]

Yes
?- replace(2,3,[1,2,3,4,2,3,2],L).

L = [1, 3, 3, 4, 3, 3, 3]

Yes
11.Hàm deleteall xoá tất cả các phần tử có giá trị bằng x trong danh sách L

```
deleteall(X,[],[]).
deleteall(X,[X|L],L1):-deleteall(X,L,L1).
deleteall(X,[Y|L],[Y|L1]):-deleteall(X,L,L1).
?- deleteall(2,[1,2,3,4,2,5,2,6,7,2],L).
```

L = [1, 3, 4, 5, 6, 7]

Yes
?- deleteall(2,[],L)
| .

```
L = []

Yes
?-  delete(2,[2],L).

No
?- deleteall(2,[2],L).

L = []

Yes
?- deleteall(2,[1,3,4,5,6],L).
L = [1, 3, 4, 5, 6]

Yes
?- deleteall(2,[1,2,3,2,5,2,6],[1,3,5,6]).

Yes
```
12.Hàm countdivk đếm các phần tử trong danh sách mà chia hết cho k
```
countdivk(K,[],0).
countdivk(K,[X|L],N):-X mod K =:=0,
                      countdivk(K,L,M),
                      N is M+1.
countdivk(K,[X|L],N):-X mod K =\=0,
                      countdivk(K,L,N).
?- countdivk(2,[],N).

N = 0

Yes
?- countdivk(2,[2],N).

N = 1

Yes
?- countdivk(2,[1,3,5,7],N).

N = 0

Yes
?- countdivk(2,[1,2,3,4,5,6],N).

N = 3

Yes
```
13.Hàm oddsum tính tổng các số lẻ trong danh sách L
```
oddsum([],0).
oddsum([X|L],N):-X mod 2 =:=1,
                 oddsum(L,M),
                 N is M+X.
```

```
oddsum([X|L],N):-X mod 2 =:=0,
                 oddsum(L,N).
?- oddsum([],N).

N = 0

Yes
?- oddsum([2,4,6],N).

N = 0

Yes
?- oddsum([1,2,3,4,5],N).

N = 9

Yes
?- oddsum([2,3,4,5,6,7,8],N).

N = 15
Yes
?- oddsum([1,2,3,4,5],9).

Yes
```

14.Hàm evencount đếm tất cả các số chẵn có trong danh sách L

```
evencount([],0).
evencount([X|L],N):-X mod 2 =:=0,
                     evencount(L,M),
                     N is M+1.
evencount([X|L],N):-X mod 2 =\=0,
                     evencount(L,N).
?- evencount([],N).

N = 0

Yes
?- evencount([1,3,5,7],N).

N = 0

Yes
?- evencount([1,2,3,4,5],N).

N = 2
?- evencount([1,2,3,2,4,6,8],N).

N = 5

Yes
?- evencount([1,2,3,4,5,6],3).
```

Yes
?- evencount([2,4,6,3],2).

No

15.Hàm Fibonacci

```
fib(0,1).
fib(1,1).
fib(N,F):-N>1,
        N1 is N-1,
        N2 is N-2,
        fib(N1,F1),
        fib(N2,F2),
        F is F1+F2.
```

?- fib(1,F).

F = 1

Yes
?- fib(0,F).

F = 1

Yes
?- fib(3,N).

N = 3

Yes
?- fib(5,B).
B = 8

Yes
?- fib(5,8).

Yes
?- fib(5,7).

No

16.Hàm fac tính n!

```
fac(0,1).
fac(N,F):-N>0,
      M is N-1,
      fac(M,F1),
      F is N*F1.
```

?- fac(0,N).

N = 1

Yes
?- fac(1,N).

N = 1

Yes
?- fac(2,N).

N = 2

Yes
?- fac(3,N).
N = 6

Yes
?- fac(5,N).

N = 120

Yes
?- fac(7,N).

N = 5040

Yes
?- fac(5,120).

Yes
?-
| fac(5,10).

No

17.Hàm power tính a^n

```
power(A,0,1).
power(A,N,P):-N>0,
            M is N-1,
            power(A,M,P1),
            P is P1*A.
```

?- power(2,0,N).

N = 1

Yes
?- power(2,1,N).

N = 2

Yes
?- power(2,3,N).
N = 8

Yes

```
?- power(2,6,N).

N = 64

Yes
?- power(2,N,64).
ERROR: Arguments are not sufficiently instantiated
  Exception: (7) _G217>0 ? creep
?- power(A,3,8).
ERROR: Arguments are not sufficiently instantiated
^  Exception: (9) _G281 is 1*_G210 ? creep
?- power(2,3,8).

Yes
?- power(2,3,4).No
```

18.Hàm c tính định thức C(n,k)= $\begin{cases} 1 \text{ nếu } k=0 \text{ hoặc } k=n \\ C(n-1,k)+C(n-1,k), \text{ với } k \text{ khác} \end{cases}$

```
c(N,K,F):-N<K,
        write('loi').
c(N,0,1).
c(N,N,1).
c(N,K,F):-N1 is N-1,
        K1 is K-1,
        c(N1,K1,F1),
        c(N1,K,F2),
        F is F1+F2.
|    c(1,2,F).
loi

F = _G188

Yes
?- c(2,0,F).

F = 1

Yes
?- c(2,2,F).

F = 1

Yes
?- c(3,2,F).

F = 3
Yes
?- c(5,2,F).

F = 10
```

Yes
?- c(5,2,10).

Yes
?- c(5,2,3).

ERROR: Arguments are not sufficiently instantiated
loi^ Exception: (10) _G311 is 1+_G299 ? creep
?- c(N,3,10).
ERROR: Arguments are not sufficiently instantiated
  Exception: (6) c(_G192, 3, 10) ? creep
?- c(5,K,10).
ERROR: Arguments are not sufficiently instantiated
  Exception: (6) c(5, _G193, 10) ? creep
?- c(5,3,10).

Yes
19.Hàm tính tổng các số từ 1 đến N
```
tong(0,0).
tong(N,S):-M is N-1,
          tong(M,S1),
          S is N+S1.
|    tong(0,N).
```

N = 0

Yes
?- tong(5,N).

N = 15

Yes
?- tong(5,15).

Yes
?- tong(6,15).
ERROR: Out of local stack
20.Hàm gcd tìm ước số chung lớn nhất của hai số
Cách 1:
```
gcd(A,B,GCD):-A=B,GCD=A.
gcd(A,B,GCD):-A<B,
          NB is B-A,
          gcd(A,NB,GCD).
gcd(A,B,GCD):-A>B,
          NA is A-B,
          gcd(NA,B,GCD).
```
?- gcd(3,5,N).

N = 1
Yes
?- gcd(4,6,N).

```
N = 2

Yes
?- gcd(8,9,N).

N = 1

Yes
?- gcd(5,15,N).

N = 5

Yes
```
Cách 2:
```
ucln(A,0,A).
ucln(A,B,N):-A<0,
        NA is -A,
        ucln(NA,B,N).
ucln(A,B,N):-B<0,
            NB is -B,
            ucln(A,NB,N).
ucln(A,B,N):-A<B,
            ucln(B,A,N).
ucln(A,B,N):-D is (A mod B),
            ucln(B,D,N).
?- ucln(4,5,N).

N = 1
Yes
?- ucln(6,8,N).

N = 2
Yes
?- ucln(4,8,N).

N = 4
Yes
?- ucln(4,8,4).

Yes
?- ucln(4,8,7).
ERROR: mod/2: Arithmetic: evaluation error: `zero_divisor'
^  Exception: (9) _G262 is 4 mod 0 ? creep
   Exception: (6) ucln(4, 8, 7) ? creep
?- ucln(3,4,1).

Yes
?- ucln(3,4,2).
ERROR: mod/2: Arithmetic: evaluation error: `zero_divisor'
^  Exception: (10) _G265 is 1 mod 0 ? creep
   Exception: (6) ucln(3, 4, 2) ? creep
```

21.Hàm sumlist tính tổng các phần tử của một danh sách.

```
sumlist([],0).
sumlist([X|L],N):-sumlist(L,M),
                  N is M+X.
?- sumlist([],N).

N = 0

Yes
?- sumlist([],0).

Yes
?- sumlist([2],N).

N = 2

Yes
?- sumlist([2,3,4,5],N).

N = 14

Yes
?- sumlist([1,2,3,4,5],15).

Yes
?- sumlist([2,5,3],2).

No
```

22.Hàm subset là hàm kiểm tra tập hợp này có phải là tập con của tập kia.

```
subset([],L).
subset([X|T],L):-member(X,L),
                 subset(T,L).
?- subset([],[]).

Yes
?- subset([],[1,2,3]).

Yes
?- subset([1,2,3],[]).

No
?- subset([1,2],[4,2,5,6,1]).

Yes
?- subset([1,2],[3,4,5,6,1]).

No
?- subset(L,[1,2,3,4,5,6]).

L = []
```

Yes
?- subset(L,[1,2,3,4,5,6]).

L = [] ;

No
23.Hàm sumlist kiểm tra danh sách này có phải là danh sách con của danh sách kia không.
concat([],L,L).
concat([H|T],L,[H|L1]):-concat(T,L,L1).
sublist(T,L):-concat(L1,L2,L),concat(T,L3,L2).
?- sublist([],[]).

Yes
?- sublist([],[1,2,3]).

Yes
?- sublist([1,2,3],[]).

No
?- sublist([1,2,3],[1,2,4,5,3]).

No
?- sublist([1,2,3],[1,2,4,5,1,2,3]).

Yes
?- sublist([1,2,3],L).

L = [1, 2, 3|_G318]

Yes
?- sublist([1,2,3],L).

L = [1, 2, 3|_G318] ;
L = [_G311, 1, 2, 3|_G324] ;

L = [_G311, _G317, 1, 2, 3|_G330] ;

L = [_G311, _G317, _G323, 1, 2, 3|_G336] ;

L = [_G311, _G317, _G323, _G329, 1, 2, 3|_G342] ;

L = [_G311, _G317, _G323, _G329, _G335, 1, 2, 3|_G348] ;

L = [_G311, _G317, _G323, _G329, _G335, _G341, 1, 2, 3|...] ;

L = [_G311, _G317, _G323, _G329, _G335, _G341, _G347, 1, 2|...] ;

L = [_G311, _G317, _G323, _G329, _G335, _G341, _G347, _G353, 1|...] ;

L = [_G311, _G317, _G323, _G329, _G335, _G341, _G347, _G353, _G359|...] ;

L = [_G311, _G317, _G323, _G329, _G335, _G341, _G347, _G353, _G359|...]

Yes
?- sublist(L,[1,2,3,4]).
L = [] ;

L = [1] ;

L = [1, 2] ;

L = [1, 2, 3] ;

L = [1, 2, 3, 4] ;

L = [] ;

L = [2] ;

L = [2, 3] ;

L = [2, 3, 4] ;

L = [] ;

L = [3] ;

L = [3, 4] ;
L = [] ;

L = [4] ;

L = [] ;

No
24.Hàm sequence kiểm tra xem
sequence([],L).
sequence([X|T],[X|L]):-sequence(T,L).
?- sequence([],[]).

Yes
?- sequen([],[1,2,3]).
ERROR: Undefined procedure: sequen/2
?- sequence([],[1,2]).

Yes
?- sequence([1,2],[2,3,1]).

No
?- sequence([1,2],[3,4,1,2]).

No

?- sequence([1,2],[1,2,3,4]).

Yes
?- sequence([1,2],[1,2]).
Yes
?- sequence([1,2],[1,2,3]).

Yes
?- sequence([1,2],[1,1,2]).

No
25.Hàm countk đếm số phần tử k trong danh sách L(k là ký tự hoặc chữ số)

```
countk(K,[],0).
countk(K,[K|L],N):-countk(K,L,M),
                    N is M+1.
countk(K,[X|L],N):-countk(K,L,N).
          ?- countk(a,[a,b,c,d],N).
```

N = 1

Yes
?- countk(a,[a,b,c,d,a,b,c,d],N).

N = 2

Yes
?- countk(2,[1,2,3,4,5,2,5,2],N).

N = 3

Yes
Tìm đường đi giữa hai đỉnh của đồ thị không đinh hướng

```
arc(a, b).
arc(a, d).
arc(b, c).
arc(b, j).
arc(c, e).
arc(d, c).
arc(e, f).
arc(e, g).
arc(g, h).
arc(i, e).
/* arc(c, a). --> cycle */
path(X, X).
path(X, Y):-
        arc(X, Z),
        arc(Z, Y).
pathc(X, Y):-
        cycle_path(X, Y, []).

/* cycle_path 4 */
```

```
cycle_path(X, X, _).
cycle_path(X, Y, L):-
        arc(X, Z),
        (not belong(Z, L)),
        cycle_path(Z, Y, [Z|L]].

/* non_oriented_path 4 */
non_oriented_path(X, Y):-
        non_oriented_cycle_path(X, Y, [X]).

/* non_oriented_cycle_path 4 */
non_oriented_cycle_path(X, X, L):-
        write(L),
        nl.

non_oriented_cycle_path(X, Y, L):-
        arc(X, Z),
        (not belong(Z, L)),
        non_oriented_cycle_path(Z, Y, [Z|L]).

non_oriented_cycle_path(X, Y, L):-
        arc(X, Z),
        (not belong(Z, L)),
        non_oriented_cycle_path(Z, Y, [Z|L]).
belong(X, [X|Q]).
belong(X, [T|Q]):-
        belong(X, Q).
path1(X, X, [X]).
path1(X, Y, L):-
        arc(X, Z),
        path1(Z, Y, L2),
        L=[X|L2].

graphe([a, b, c, d, e, f, g, h, i, j]).
exist_cycle(X, Y, L):-
        arc(X, Y).
exist_cycle(X, Y, L):-
        arc(X, Z).
        (not belong(Z, L)),
        exist_cycle(Z, Y, [Z|L]).

verif_cycle([T|Q]):-
        exist_cycle(T,T, []);verif_cycle(Q).
verif_pas_cycle([T|Q]):-
        not(verif_cycle([T|Q])).
distance(X, X, 0).
distance(X, Y, D):-
        arc(X, Z),
        distance(Z, Y, D1),
        D is D1 + 1.
```

```
/* nhap */
display:-
        write('Chuong trinh tim duong di giua hai dinh cua do thi.'),
        nl,
        write('Nhap hai dinh cua do thi: '),
        nl,
        print('X: '),
        read(X),
        write('Y: '),

        read(Y),
        write('Duong di tu dinh '),write(X),write(' den dinh '),write(Y),
        write(' qua cac dinh: '),/* write(X), */
        distance(X,Y,D,L),
        /*write(Y). */
        nl,
        write('Tu dinh '),write(X),write(' den dinh '),
        write(Y),write(' phai qua '),write(D),write(' buoc').

        /*write('List '),
        write(L).*/
```