רשימות

דוגמאות ראשונות

מהי רשימה ?

- רשימה – מבנה נתונים המכיל סדרת איברים .

[1,2,33,43,8,11,house,3,roy]

- נגדיר רשימה **בצורה רקורסיבית** .
  - רשימה יכולה להיות ריקה [ ] .
  - או סדרת איברים הנתנת לתיאור בעזרת חלוקה לאיברים מובילים ורשימת שארית :

[1,4,22] = [1|[4,22]] = [1| [4|[22]]] = [1| [4|[22|[ ]]]]

- ההגדרה הרקורסיבית תאפשר לנו לזהות ולאפיין הרכב רשימות .

[1,4,22] = [1|[4,22]] = [1,4| [22]]

- בספר מתוארת ההגדרה הרקורסיבית המקורית :

Example:

.(a,[]) = [a|[]] = [a]

.(a,.(b,[])) = [a|[b|[]]] = [a,b]

.(a,.(b,.(c,[])) = [a|[b|[c|[]]]] = [a,b,c]

- בד"כ לא נשתמש בהגדרה זו .

כמה דוגמאות :

- שיכות לרשימה :

```
member(X,[X|Xs]).
member(X,[Y|Ys]):-member(X,Ys).
```

ניתן לבצע שאילתות שונות על החוק :

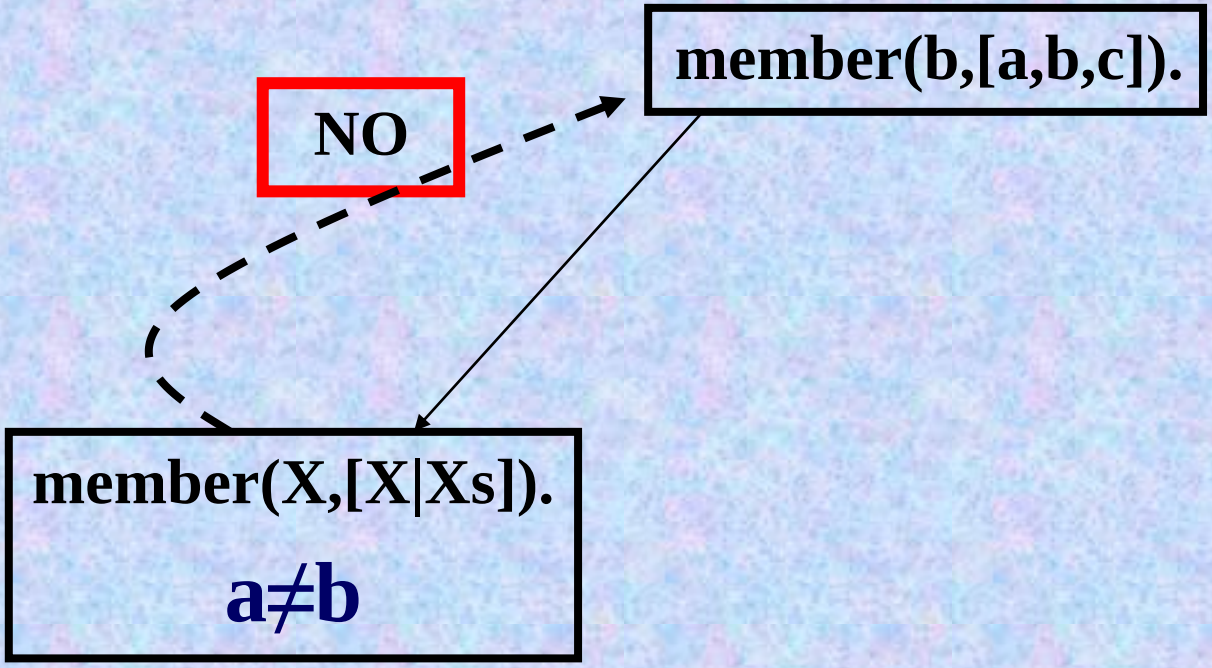
```
?- member(a,[a,b,c]).
?- member(d,[a,b,c]).
?- member(X,[a,b,c]).
?- member(b,X)(!).
```

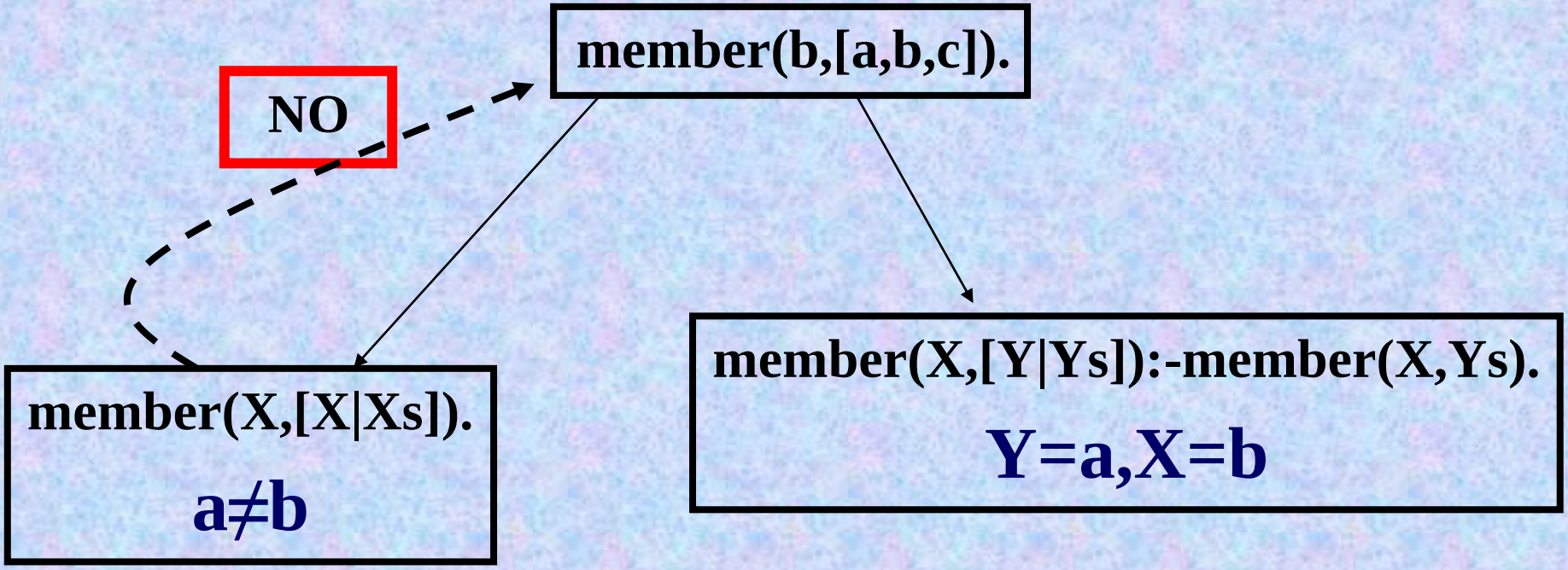
מה יהיו תוצאות ההרצה
של השאילתות הנ"ל ?

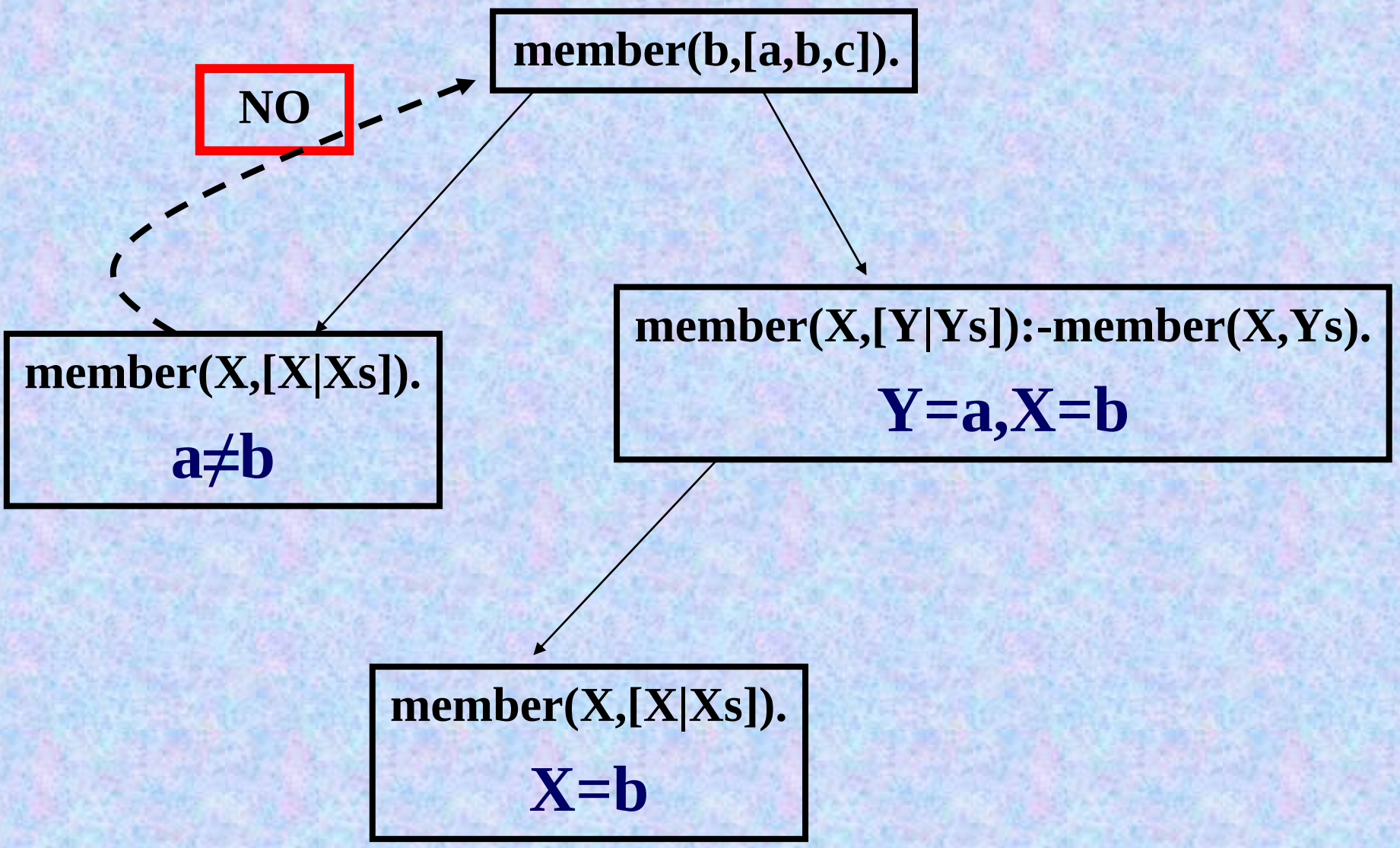
member(b,[a,b,c]).

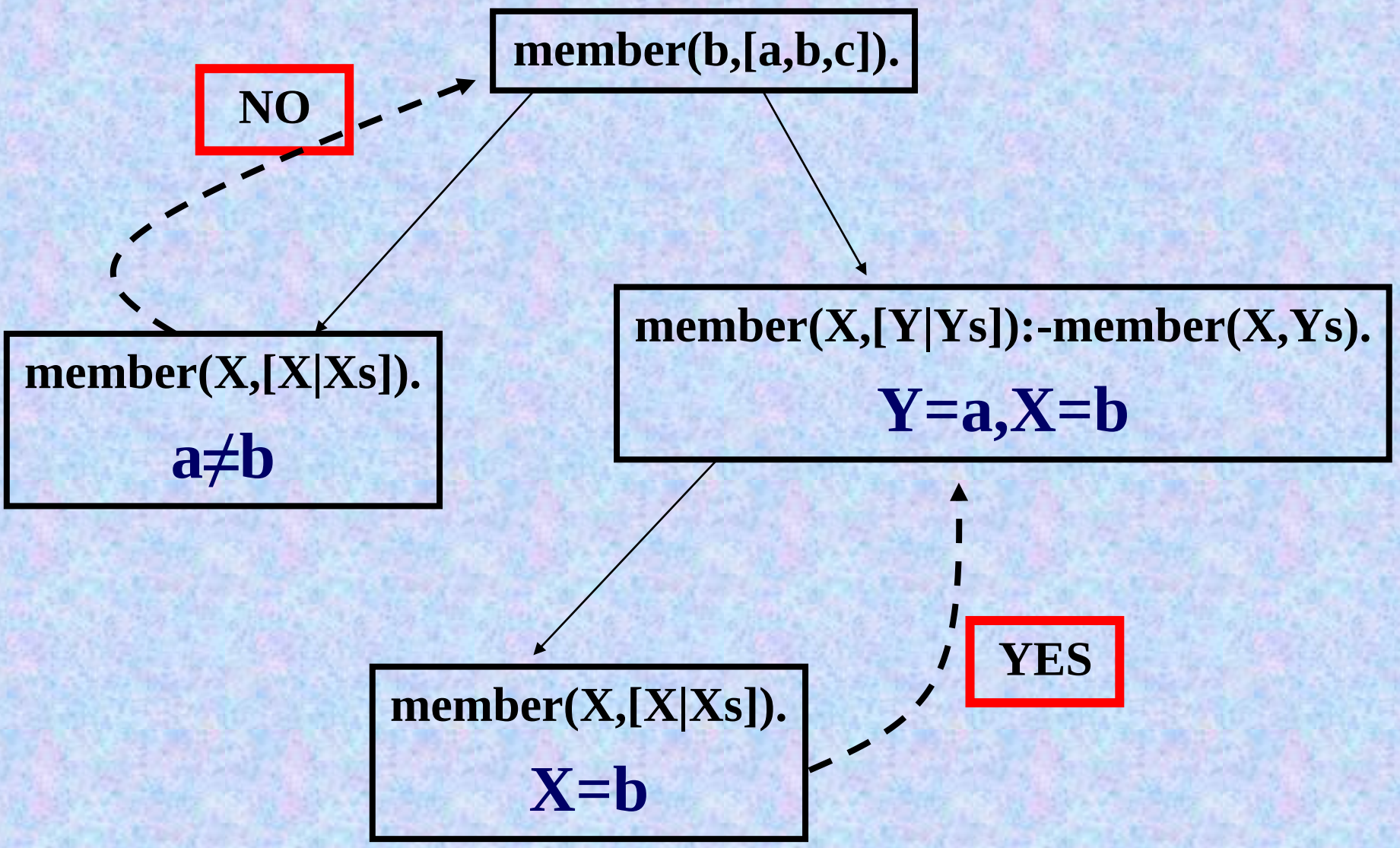
member(b,[a,b,c]).

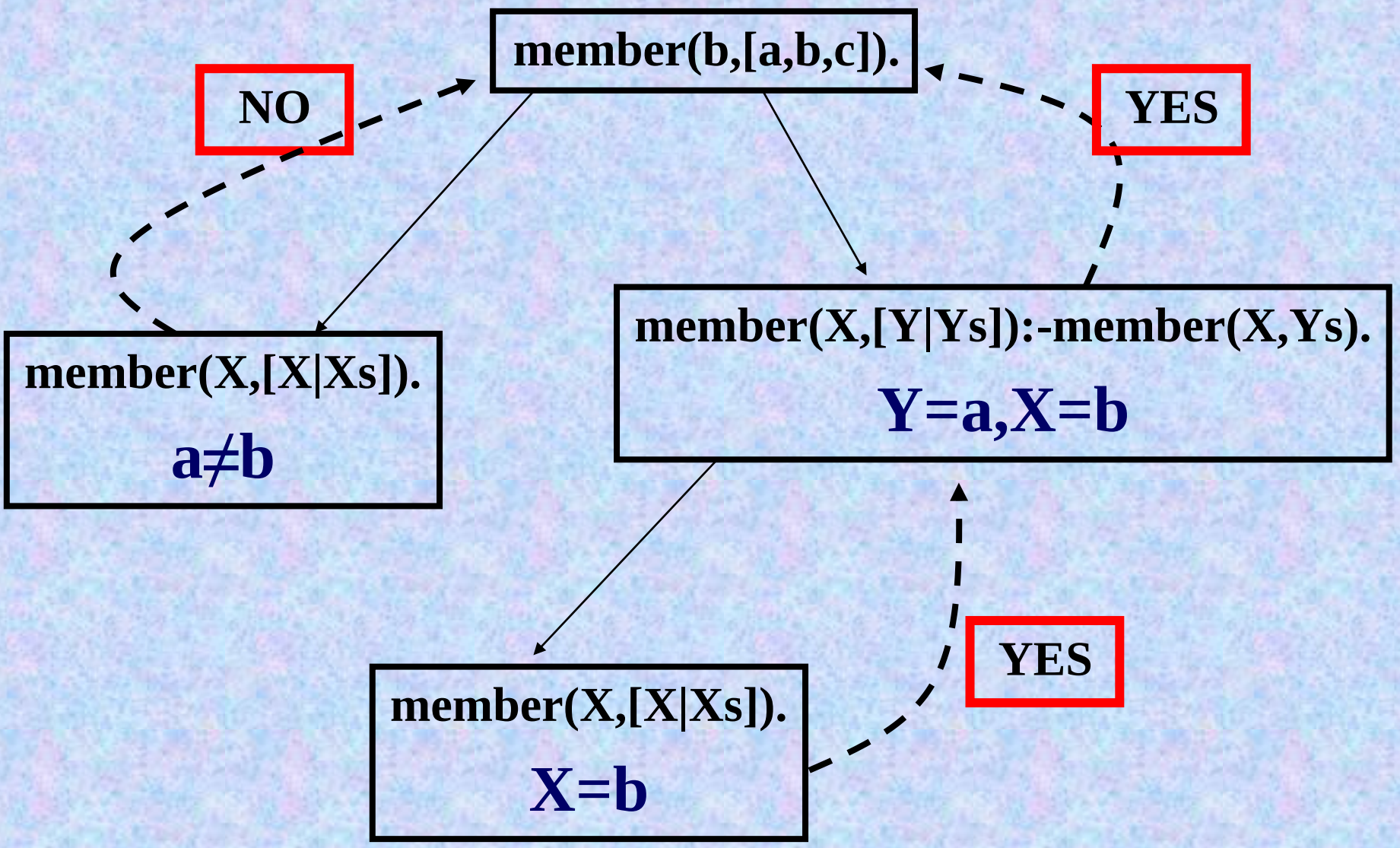
member(X,[X|Xs]).


member(b,[a,b,c]).
NO
member(X,[X|Xs]).
a≠b


member(b,[a,b,c]).
NO
member(X,[X|Xs]).
a≠b
member(X,[Y|Ys]):-member(X,Ys).
Y=a,X=b


member(b,[a,b,c]).
NO
member(X,[X|Xs]).
a≠b
member(X,[Y|Ys]):-member(X,Ys).
Y=a,X=b
member(X,[X|Xs]).
X=b


member(b,[a,b,c]).
NO
member(X,[X|Xs]).
a≠b
member(X,[Y|Ys]):-member(X,Ys).
Y=a,X=b
YES
member(X,[X|Xs]).
X=b



והתשובה שנקבל תהיה YES

```
member(X,[X|_]).
member(X,[_|Ys]):-
    member(X,Ys).
```

- אי שייכות לרשימה :

```
nonmember(X,[Y|Ys]) :-
        X\=Y,
        nonmember(X,Ys).
nonmember(X,[]).
```

באילו דרכים נוספות ניתן
לממש פרדיקט זה ?

- סכימת איברים ברשימה :

```
sum([I|Is],S):-
   sum(Is,S0),
   S is S0 + I.
sum([ ],0).
```

```
?- sum([3,6,2,-1],Res).
```

?- sum([3,6,2,-1],Res).

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = 3 Is = [6,2,-1]

?- sum([3,6,2,-1],Res).

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = 3 Is = [6,2,-1]

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = 6 Is = [2,-1]

?- sum([3,6,2,-1],Res).

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = 3 Is = [6,2,-1]

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = 6 Is = [2,-1]

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = 2 Is = [-1]

?- sum([3,6,2,-1],Res).

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = 3 Is = [6,2,-1]

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = 6 Is = [2,-1]

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = 2 Is = [-1]

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = -1 Is = []

?- sum([3,6,2,-1],Res).

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = 3 Is = [6,2,-1]

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = 6 Is = [2,-1]

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = 2 Is = [-1]

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = -1 Is = []

sum([],0).

?- sum([3,6,2,-1],Res).

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = 3 Is = [6,2,-1]

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = 6 Is = [2,-1]

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = 2 Is = [-1]

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = -1 Is = []

sum([],0).

S = -1

?- sum([3,6,2,-1],Res).

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = 3 Is = [6,2,-1]

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = 6 Is = [2,-1]

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = 2 Is = [-1]

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = -1 Is = []

sum([],0).

S = -1

S = 1

?- sum([3,6,2,-1],Res).

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = 3 Is = [6,2,-1]

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = 6 Is = [2,-1]

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = 2 Is = [-1]

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = -1 Is = []

sum([],0).

S = -1

S = 1

S = 7

?- sum([3,6,2,-1],Res).

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = 3 Is = [6,2,-1]

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = 6 Is = [2,-1]

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = 2 Is = [-1]

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = -1 Is = []

sum([],0).

S = -1

S = 1

S = 7

?- sum([3,6,2,-1],Res).

ונקבל :

Res = 10

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = 3 Is = [6,2,-1]

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = 6 Is = [2,-1]

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = 2 Is = [-1]

sum([I|Is],S):- sum(Is,S0), S is S0 + I.

I = -1 Is = []

sum([],0).

S = -1

S = 1

S = 7

עוד דרך לסכום את איברי הרשימה :

```
sum(List,Sum):-   sum(List,0,Sum).
sum([],Sum,Sum).
sum([X|Xs],S0,Sum):-
   S1 is X + S0 ,
   sum(Xs,S1,Sum).
```

עמדו על ההבדלים !

```
?- sum([3,6,2,-1],0,Res).
```

?- sum([3,6,2,-1],0,Res).

sum([X|Xs],S0,Sum):-S1 is X + S0,sum(Xs,S1,Sum).

X = 3 Xs = [6,2,-1] ➔ S1 = 0+3

?- sum([3,6,2,-1],0,Res).

sum([X|Xs],S0,Sum):-S1 is X + S0,sum(Xs,S1,Sum).

X = 3 Xs = [6,2,-1] ➔ S1 = 0+3

sum([X|Xs],3,Sum):-S1 is X + S0 , sum(Xs,S1,Sum).

X = 6 Xs = [2,-1] ➔ S1 = 6+3

?- sum([3,6,2,-1],0,Res).

sum([X|Xs],S0,Sum):-S1 is X + S0,sum(Xs,S1,Sum).

X = 3 Xs = [6,2,-1] ➔ S1 = 0+3

sum([X|Xs],3,Sum):-S1 is X + S0 , sum(Xs,S1,Sum).

X = 6 Xs = [2,-1] ➔ S1 = 6+3

sum([X|Xs],S0,Sum):-S1 is X + S0 , sum(Xs,S1,Sum).

X = 2 Xs = [-1] ➔ S1 = 9+2

?- sum([3,6,2,-1],0,Res).

sum([X|Xs],S0,Sum):-S1 is X + S0,sum(Xs,S1,Sum).

X = 3 Xs = [6,2,-1] ➔ S1 = 0+3

sum([X|Xs],3,Sum):-S1 is X + S0 , sum(Xs,S1,Sum).

X = 6 Xs = [2,-1] ➔ S1 = 6+3

sum([X|Xs],S0,Sum):-S1 is X + S0 , sum(Xs,S1,Sum).

X = 2 Xs = [-1] ➔ S1 = 9+2

sum([X|Xs],S0,Sum):-S1 is X + S0,sum(Xs,S1,Sum).

X = -1 Xs = [ ] ➔ S1 = 11+(-1)

?- sum([3,6,2,-1],0,Res).

sum([X|Xs],S0,Sum):-S1 is X + S0,sum(Xs,S1,Sum).

X = 3 Xs = [6,2,-1] ➔ S1 = 0+3

sum([X|Xs],3,Sum):-S1 is X + S0 , sum(Xs,S1,Sum).

X = 6 Xs = [2,-1] ➔ S1 = 6+3

sum([X|Xs],S0,Sum):-S1 is X + S0 , sum(Xs,S1,Sum).

X = 2 Xs = [-1] ➔ S1 = 9+2

sum([X|Xs],S0,Sum):-S1 is X + S0,sum(Xs,S1,Sum).

X = -1 Xs = [ ] ➔ S1 = 11+(-1)

sum([],Sum,Sum).

?- sum([3,6,2,-1],0,Res).

sum([X|Xs],S0,Sum):-S1 is X + S0,sum(Xs,S1,Sum).

X = 3 Xs = [6,2,-1] ➔ S1 = 0+3

sum([X|Xs],3,Sum):-S1 is X + S0 , sum(Xs,S1,Sum).

X = 6 Xs = [2,-1] ➔ S1 = 6+3

sum([X|Xs],S0,Sum):-S1 is X + S0 , sum(Xs,S1,Sum).

X = 2 Xs = [-1] ➔ S1 = 9+2

sum([X|Xs],S0,Sum):-S1 is X + S0,sum(Xs,S1,Sum).

X = -1 Xs = [ ] ➔ S1 = 11+(-1)

sum([],Sum,Sum).

Sum = 10

?- sum([3,6,2,-1],0,Res).

sum([X|Xs],S0,Sum):-S1 is X + S0,sum(Xs,S1,Sum).

X = 3 Xs = [6,2,-1] ➔ S1 = 0+3

sum([X|Xs],3,Sum):-S1 is X + S0 , sum(Xs,S1,Sum).

X = 6 Xs = [2,-1] ➔ S1 = 6+3

sum([X|Xs],S0,Sum):-S1 is X + S0 , sum(Xs,S1,Sum).

X = 2 Xs = [-1] ➔ S1 = 9+2

sum([X|Xs],S0,Sum):-S1 is X + S0,sum(Xs,S1,Sum).

X = -1 Xs = [ ] ➔ S1 = 11+(-1)

sum([],Sum,Sum).

Sum = 10

Sum = 10

?- sum([3,6,2,-1],0,Res).

sum([X|Xs],S0,Sum):-S1 is X + S0,sum(Xs,S1,Sum).

X = 3 Xs = [6,2,-1] ➔ S1 = 0+3

sum([X|Xs],3,Sum):-S1 is X + S0 , sum(Xs,S1,Sum).

X = 6 Xs = [2,-1] ➔ S1 = 6+3

sum([X|Xs],S0,Sum):-S1 is X + S0 , sum(Xs,S1,Sum).

X = 2 Xs = [-1] ➔ S1 = 9+2

sum([X|Xs],S0,Sum):-S1 is X + S0,sum(Xs,S1,Sum).

X = -1 Xs = [ ] ➔ S1 = 11+(-1)

sum([],Sum,Sum).

Sum = 10

Sum = 10

Sum = 10

?- sum([3,6,2,-1],0,Res).

sum([X|Xs],S0,Sum):-S1 is X + S0,sum(Xs,S1,Sum).

X = 3 Xs = [6,2,-1] ➔ S1 = 0+3

sum([X|Xs],3,Sum):-S1 is X + S0 , sum(Xs,S1,Sum).

X = 6 Xs = [2,-1] ➔ S1 = 6+3

sum([X|Xs],S0,Sum):-S1 is X + S0 , sum(Xs,S1,Sum).

X = 2 Xs = [-1] ➔ S1 = 9+2

sum([X|Xs],S0,Sum):-S1 is X + S0,sum(Xs,S1,Sum).

X = -1 Xs = [ ] ➔ S1 = 11+(-1)

sum([],Sum,Sum).

Sum = 10

Sum = 10

Sum = 10

Sum = 10

?- sum([3,6,2,-1],0,Res).

sum([X|Xs],S0,Sum):-S1 is X + S0,sum(Xs,S1,Sum).

X = 3 Xs = [6,2,-1] ➔ S1 = 0+3

sum([X|Xs],3,Sum):-S1 is X + S0 , sum(Xs,S1,Sum).

X = 6 Xs = [2,-1] ➔ S1 = 6+3

sum([X|Xs],S0,Sum):-S1 is X + S0 , sum(Xs,S1,Sum).

X = 2 Xs = [-1] ➔ S1 = 9+2

sum([X|Xs],S0,Sum):-S1 is X + S0,sum(Xs,S1,Sum).

X = -1 Xs = [ ] ➔ S1 = 11+(-1)

sum([],Sum,Sum).

Sum = 10

Sum = 10

Sum = 10

Sum = 10

Sum = 10

?- sum([3,6,2,-1],0,Res).

ונקבל :

Res = 10

sum([X|Xs],S0,Sum):-S1 is X + S0,sum(Xs,S1,Sum).

X = 3 Xs = [6,2,-1] ➔ S1 = 0+3

sum([X|Xs],3,Sum):-S1 is X + S0 , sum(Xs,S1,Sum).

X = 6 Xs = [2,-1] ➔ S1 = 6+3

sum([X|Xs],S0,Sum):-S1 is X + S0 , sum(Xs,S1,Sum).

X = 2 Xs = [-1] ➔ S1 = 9+2

sum([X|Xs],S0,Sum):-S1 is X + S0,sum(Xs,S1,Sum).

X = -1 Xs = [ ] ➔ S1 = 11+(-1)

sum([],Sum,Sum).

Sum = 10

Sum = 10

Sum = 10

Sum = 10

Sum = 10

- מציאת אורך רשימה :

```
length([],0).
length([X|Xs],N):-
  N>0,
  N1 is N-1,
  length(Xs,N1).
```

מה יקרה אם לא נבדוק את N ?

- גרסה זאת של הפתרון יכולה לזהות רק אם הפרמטר השני הוא באמת אורך הרשימה .

```
length([a,a,a,a,a],5).
yes
```

- מציאת אורך רשימה :

```
length([],0).
length([X|Xs],N):-
   length (Xs,N1),
   N is N1+1.
```

גרסה זאת יכולה גם לזהות התאמה לאורך וגם למנות את אורך הרשימה .

```
?- length(LongList,SmallN).
length1([a,a,a,a,a],X).
X = 5
```

מציאת אורך רשימה (3) :

```
length(L,N):-
    length(L,0,N).
length([],N,N).
length([X|Xs],M,N):-
      M1 is M+1,
      length(Xs,M1,N).
```

שימו לה לשימוש בצובר, האם זה מייעל את הפרדיקט ?

- שירשור שתי רשימות :

```
conc([],Ys,Ys).
conc([X|Xs],Ys,[X|Zs]):-conc(Xs,Ys,Zs).
```

- זהו פרדיקט שימושי ביותר !!
- שימו לב איך נבנית הרשימה ...

```
?- conc([3,roy],[tal,6,7],R).
```

?- conc([3,roy],[tal,6,7],R).

conc([X|Xs],Ys,[X|Zs]):- conc(Xs,Ys,Zs).

X = 3 Xs = [roy] Ys = [tal,6,7] ➔ Res = [3|Zs]

?- conc([3,roy],[tal,6,7],R).

conc([X|Xs],Ys,[X|Zs]):- conc(Xs,Ys,Zs).

X = 3 Xs = [roy] Ys = [tal,6,7] ➔ Res = [3|Zs]

conc([X|Xs],Ys,[X|Zs]):-conc(Xs,Ys,Zs).

X = roy Xs = [] Ys = [tal,6,7] ➔ Res = [roy|Zs]

?- conc([3,roy],[tal,6,7],R).

conc([X|Xs],Ys,[X|Zs]):- conc(Xs,Ys,Zs).

X = 3 Xs = [roy] Ys = [tal,6,7] ➔ Res = [3|Zs]

conc([X|Xs],Ys,[X|Zs]):-conc(Xs,Ys,Zs).

X = roy Xs = [] Ys = [tal,6,7] ➔ Res = [roy|Zs]

conc([],Ys,Ys).

Ys = [tal,6,7] ➔ Res = [tal,6,7]

?- conc([3,roy],[tal,6,7],R).

conc([X|Xs],Ys,[X|Zs]):- conc(Xs,Ys,Zs).

X = 3 Xs = [roy] Ys = [tal,6,7] ➔ Res = [3|Zs]

conc([X|Xs],Ys,[X|Zs]):-conc(Xs,Ys,Zs).

X = roy Xs = [] Ys = [tal,6,7] ➔ Res = [roy|Zs]

conc([],Ys,Ys).

Ys = [tal,6,7] ➔ Res = [tal,6,7]

?- conc([3,roy],[tal,6,7],R).

conc([X|Xs],Ys,[X|Zs]):- conc(Xs,Ys,Zs).

X = 3 Xs = [roy] Ys = [tal,6,7] ➔ Res = [3|Zs]

conc([X|Xs],Ys,[X|Zs]):-conc(Xs,Ys,Zs).

X = roy Xs = [] Ys = [tal,6,7] ➔ Res = [roy|Zs]

[tal,6,7]

conc([],Ys,Ys).

Ys = [tal,6,7] ➔ Res = [tal,6,7]

?- conc([3,roy],[tal,6,7],R).

conc([X|Xs],Ys,[X|Zs]):- conc(Xs,Ys,Zs).

X = 3 Xs = [roy] Ys = [tal,6,7] ➔ Res = [3|Zs]

conc([X|Xs],Ys,[X|Zs]):-conc(Xs,Ys,Zs).

X = roy Xs = [] Ys = [tal,6,7] ➔ Res = [roy|Zs]

[tal,6,7]

conc([],Ys,Ys).

Ys = [tal,6,7] ➔ Res = [tal,6,7]

?- conc([3,roy],[tal,6,7],R).

conc([X|Xs],Ys,[X|Zs]):- conc(Xs,Ys,Zs).

X = 3 Xs = [roy] Ys = [tal,6,7] ➔ Res = [3|Zs]

[roy,tal,6,7]

conc([X|Xs],Ys,[X|Zs]):-conc(Xs,Ys,Zs).

X = roy Xs = [] Ys = [tal,6,7] ➔ Res = [roy|Zs]

[tal,6,7]

conc([],Ys,Ys).

Ys = [tal,6,7] ➔ Res = [tal,6,7]

?- conc([3,roy],[tal,6,7],R).

ונקבל :

Res = *[3,roy,tal,6,7]*

conc([X|Xs],Ys,[X|Zs]):- conc(Xs,Ys,Zs).

X = 3 Xs = [roy] Ys = [tal,6,7] ➔ Res = [3|Zs]

[roy,tal,6,7]

conc([X|Xs],Ys,[X|Zs]):-conc(Xs,Ys,Zs).

X = roy Xs = [] Ys = [tal,6,7] ➔ Res = [roy|Zs]

[tal,6,7]

conc([],Ys,Ys).

Ys = [tal,6,7] ➔ Res = [tal,6,7]

החזרת רשימת כל האיברים שגדולים מ 5 :

```
bigger_than_5([],[]).
bigger_than_5([X|Xs],[X|Ys]):-
        X>5 ,
         bigger_than_5(Xs,Ys).
bigger_than_5([X|Xs],Ys):-
        X=<5 ,
         bigger_than_5(Xs,Ys).
```

החזרת כל האיברים ברשימה שגדולים מהאיבר שקודם להם :

```
smallSort([],[]).
smallSort([ _ ],[ ]).
smallSort([V1,V2|Tail],[V2|NextRes]):-
    V2 > V1 ,
    smallSort([V2|Tail],NextRes).
smallSort([V1,V2|Tail],NextRes):-
    V2 =< V1 ,
    smallSort([V2|Tail],NextRes).
```

- היפוך רשימה :

```
reverse([],[]).
reverse([X|Xs],Zs):-
   reverse(Xs,Ys), conc(Ys,[X],Zs).
```

באילו דרכים נוספות ניתן
לממש פרדיקט זה ?

- היפוך רשימה יעיל יותר בסדר גודל :

```
rev(L,Res):-
   rev(L,[],Res).
rev([X|Xs],TempList,Res):-
   rev(Xs,[X|TempList],Res).
rev([],Res,Res).
```

שימו לב לרעיון ה"צובר"
הפעם שיפור משמעותי יותר

- החזרת "אמת " אם כל האיברים ברשימה הראשונה מופיעים פעמיים במקומות מקבילים ברשימה השניה :

```
double([],[]).
double([X|Xs],[X,X|Ys]):-
        double(Xs,Ys).
```

ומה אם הסדר לא היה חשוב?

עבודה על רשימת רשימות

```
deep_member(D,L,N):-
   dm(D,L,N,1).
dm(D,[D|_],N,N).
dm(D,[X|_],N,A):-
   is_list(X),
   A1 is A+1,
   dm(D,X,N,A1).
dm(D,[_|L],N,A):-
   dm(D,L,N,A).

is_list([]).
is_list([A|T]):-
   is_list(T).
```

```
/*
|?-
deep_member(a,[a,[s,c,[d],[a,s],[]]],N).
N = 1 ;

N = 3 ;

|?-
deep_member(qq,[a,[s,c,[d],[a,s],[]]],N).
no
*/
```

העלאת כל איבר ב 1

```
inc([],[]).
inc([X|Xs],[Y|Ys]):-
    Y is X + 1 ,inc(Xs,Ys).
```

```
inc(List,Res):-inc(List,[],Res).
inc([],Res,Res).
inc([X|Xs],Temp,Res):-
    Y is X + 1 ,inc(Xs,[Y|Temp],Res).
```

?

- חלוקת הרשימה לשתי רשימות פחות או יותר שוות בארכן .

```
dividelist([X1,X2|Tail],[X1|L1],[X2|L2]):-
        dividelist(Tail,L1,L2).
dividelist([X1],[X1],[]).
dividelist([],[],[]).
```

- **מחיקת איבר מרשימה(1) :**

```
del(X,[X|Xs],Xs).
del(X,[Y|Ys],[Y|Zs]):-
   X \= Y,
  del(X,Ys,Zs).
```

```
?- del(3,[1,2,3,4],L).
?- del(3,[1,3,3,2,3,4,3],L).
```

- מחיקת איבר מרשימה (2):

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

```
del(X,[],[]).
del(X,[X|Xs],Ys):-
        del(X,Xs,Ys).
del (Z,[X|Xs], [X|Ys]):-
        X\=Z,
        del(Z, Xs,Ys).
```

מחיקת איבר מרשימה (3) :

del(X,[X|Xs],Xs).

del(X,[Y|Ys],[Y|Zs]):-
del(X,Ys,Zs).

מה קורה אם Xלא מופיע
ברשימה במימוש זה ?

?- del(3,[1,2,3,4],L).
?- del(X,[1,2,3,4],L).

- תת רשימה תחילית של רשימה

```
prefix([],Ys).
prefix([X|Xs],[X|Ys]) :-
   prefix(Xs,Ys).
```

- תת רשימה סופית של רשימה

```
    suffix(Xs,Xs).
suffix(Xs,[Y|Ys]) :-
suffix(Xs,Ys).
```

תת רשימה :

a: Suffix of a prefix

```
sublist(Xs,Ys) :-
   prefix(Ps,Ys), suffix(Xs,Ps).
```

שימו לב להבדלים בין המימושים השונים !!!

b: Prefix of a suffix

```
sublist(Xs,Ys) :-
        prefix(Xs,Ss), suffix(Ss,Ys).
```

c: Recursive definition of a sublist

```
sublist(Xs,Ys) :-
        prefix(Xs,Ys).
sublist(Xs,[Y|Ys]) :-
         sublist(Xs,Ys).
```

d: Prefix of a suffix, using conc

```
sublist(Xs,AsXsBs):-
   conc(As,XsBs,AsXsBs),
   conc(Xs,Bs,XsBs).
```

e: Suffix of a prefix, using conc

```
sublist(Xs,AsXsBs):-
        conc(AsXs,Bs,AsXsBs),
conc(As,Xs,AsXs).
```

- תמורה (permutation) שלרשימה :

```
permutation([],[]).
permutation(Xs,[Z|Zs]):-
        del(Z,Xs,Ys),
        permutation(Ys,Zs).
```

- מיון של רשימה :
  - (שימוש ב permutation)

```
ordered([X]).
ordered([X,Y|Rest]):-
   X=<Y,ordered([Y|Rest]).

sort(Xs,Ys):-
   permutation(Xs,Ys),  ordered(Ys).
```

מהי הסיבוכיות?

- מיון הכנסה :

```
sort([],[]).
sort([X|Xs],Ys):-
   sort(Xs,Zs),insert(X,Zs,Ys).

insert(X,[],[X]).
insert(X,[Y|Ys],[Y|Zs]):-
   X>Y, insert(X,Ys,Zs).
insert(X,[Y|Ys],[X,Y|Ys]):-
   X=<Y.
```

מהי הסיבוכיות?

```
make([],0).
make([N|Tail],N):-
   N>0,
   N1 is N - 1,make(Tail,N1).

make2(L,L,0).
make2(Temp,L,N):-
   N>0,
   N1 is N - 1,make2([N|Temp],L,N1).
```

מה ההבדל ?

מה עושה הפרדיקט הבא ?

```
what1(L,[],L).
what1([X|T],[X|T1],L):-
   what1 (T,T1,L).
```

מה עושה הפרדיקט הבא ?

```
what2(L,L1,L2):-
   append(L1,L2,L).
```

- מה עושה הפרדיקט הבא ?

```
what(X,Y,[],[] ).

what(X,Y,[X|Xs],[Y|Ys]):-
   what(X,Y,Xs,Ys).

what(X,Y,[Z|Xs],[Z|Ys]):-
        X\=Z,
        what(X,Y,Xs,Ys).
```

- מה עושה הפרדיקט הבא ?

```
what([],[]).
what([X|Xs],[X,X|Ys]):-
        what(Xs,Ys).
```

- כתוב פרדיקט המחזיר רשימה שבה הערכים 1 עד 100 .

```
make([],0).
make([N|Tail],N):-
        N>0,
        N1 is N - 1,make(Tail,N1).
```

- או בדרך אחרת :

```
make2(L,L,0).
make2(Temp,L,N):-
        N>0,
        N1 is N - 1,make2([N|Temp],L,N1).
```

תת רשימה :

- כתוב פרדיקט המקבל שתי רשימות ומצליח רק אם הרשימה הראשונה מוכלת כולה ברשימה השנייה .

- כתוב פרדיקט המקבל שתי רשימות ומצליח רק אם הרשימה הראשונה מוכלת כולה ברשימה השנייה , כשסדר מופעי איברי הרשימה הראשונה ברשימה השנייה נשמר (לא בהכרח ברצף).

רשימת רשימות :

- כתוב פרדיקט המקבל רשימה של רשימות ומחזיר רשימה אחת המכילה את כל איברי רשימה הרשימות לפי הסדר ללא כינון .
- חזור על הבעיה הקודמת ללא שימוש בשירשור .

כתוב תכנית המקבלת כקלט רשימת מספרים שלמים. התכנית תחזיר כפלט רשימה המתקבלת מרשימת הקלט באופן הבא: לאחר כל מספר טבעי ברשימת הקלט יש להוסיף את כל המספרים הטבעיים הקטנים ממנו בסדר יורד.

```
| ?- make_list([0,-8,4,-1,3,5,1],L).
L = [0,-8,4,3,2,1,-1,3,2,1,5,4,3,2,1,1]
```

```
make_list([],[]).
make_list([X|Xs],[X|Ys]):-
   X=<1,
 make_list(Xs,Ys).
   make_list([X|Xs],[X|Ys]):-
   X>1,
   Y is X-1,
 make_list([Y|Xs],Ys).
```

שאלה :

הגדר פרדיקט find_sum (List1, Sum, List2) המקבל כקלט רשימת מספרים שלמים (ללא חזרות) ומספר שלם Sum. על הפרדיקט לייצר כפלט רשימה List2 אשר תכיל איברים מתוך List1 (ללא חזרות) אשר סכומם הוא Sum.

למשל:

```
?- find_sum ([5,2,1,8,6,10], 20, List2).
   List2 = [5,1,8,6]
   List2 = [2,8,10]
   no
```

רשימת המספרים הגדולים ברשימת רשימות במעבר יחיד ! וללא שימוש ביחסי עזר !

```
|?- maxL([[2,3,44,5],[5,66,4],[5]],Res).
Res = [44,66,5] ;
```

```
maxL([[X,Y|Ys]|Tail],Rest):-
        X>Y,maxL([[X|Ys]|Tail],Rest)
        ;
        Y>=X , maxL([[Y|Ys]|Tail],Rest).


maxL([[X]|Tail],[X|Rest]):-
        maxL(Tail,Rest).


maxL([[]|Tail],Rest):-
        maxL(Tail,Rest).
maxL([],[]).
```

שאלה 5 :

כתוב תכנית sequence (List, Length, SubSeq) המקבלת כקלט רשימת מספרים ומספר טבעי Length.

התכנית תחזיר ב-SubSeq תת-רשימה של איברים המהווים סדרה עוקבת (רצופה) באורך Length.

דוגמאות:

?- sequence ([3,7,1,6,2,8,4], 4, SubSeq).

SubSeq = [1,2,3,4]

שאלה 3 :

קווים במישור מתוארים בעזרת אוסף עובדות מהצורה: seg(Point1, Point2).

כאשר נקודה מתוארת על-ידי היחס point(X, Y) (ראה סעיף 2.2 בספר הלימוד).

<u>מצולע</u> הוא עקום (דהיינו, רצף קווים שונים המחוברים זה לזה) סגור.

הנח שצלעות המצולע אינן נחתכות.

כתוב והרץ פרדיקט polygon(ListofPoints, Length) המחזיר את רשימת הנקודות המהוות את קודקודיו של מצולע כלשהו במישור וכן את מספר צלעות המצולע.(במצולע יש לפחות שלוש צלעות).התכנית תחזיר בנסיגה לאחור את **כל** המצולעים המוגדרים על-ידי העובדות שבבסיס הנתונים.הרץ את התכנית עבור העובדות שלהלן:

seg (point(2,3), point(4,1)).
seg (point(5,5), point (4,1)).
seg (point(2,3), point (5,5)).
seg (point(13,6), point (15,6)).
seg (point(12,4), point (13,2)).
seg (point(15,2), point (13,2)).
seg (point(13,6), point (12,4)).
seg (point(16,4), point (15,2)).

seg (point(16,4), point (15,6)).
seg (point(10,6), point (2,3)).
seg (point(5,9), point (12,9)).
seg (point(3,6), point (5,9)).
seg (point(3,6), point (10,6)).
seg (point(12,9), point (10,6)).
seg (point(12,9), point (13,6)).

שאלה 9:

כתוב פרדיקט find_expression(NumsList, Target, Expression) המקבל רשימת מספרים וסכום מטרה. הפרדיקט יוצר מרשימת המספרים ביטוי אריתמטי שערכו הוא סכום המטרה.

הביטוי נוצר ע"י שימוש בארבע פעולות החשבון: חיבור (+), חיסור (-), כפל (*) וחילוק (/). ניתן להשתמש בסוגריים. אין לשנות את סדר המספרים.

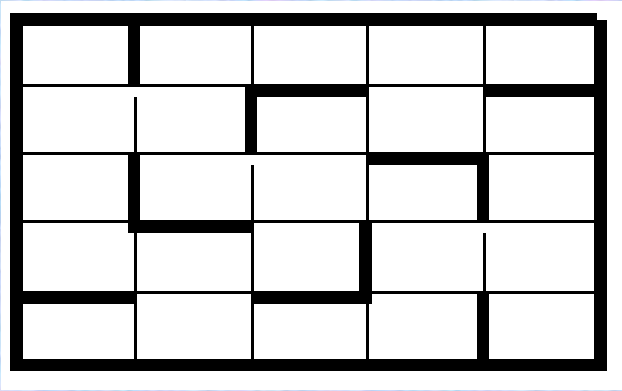
```
?-  find_expression([1,2,3,4],5,Expression).
Expression = (1 + 2) / 3 + 4;
Expression = 1 - (2 - 3) * 4;
Expression = (1 + 2) * 3 - 4
```

הדרכה: ראשית יש להחליט על מיקומו וסוגו של האופרטור הראשי, ואז להמשיך רקורסיבית עם שני האופרנדים של האופרטור הראשי. למשל, עבור הדוגמא שהובאה, בפתרון הראשון, האופרטור הראשי הוא חיבור ושני האופרנדים שלו הם: (1 + 2) / 3 ו- 4. בפתרון השני, האופרטור הראשי הוא חיסור, ושני האופרנדים שלו הם: 1 ו- (2 - 3) * 4. בפתרון השלישי, האופרטור הראשי הוא חיסור, ושני האופרנדים שלו הם: (1 + 2) * 3 ו- 4. שימו לב, בדרך זו אין כל צורך להתעסק בסוגריים, ופרולוג יוסיף את הסוגריים, במקום בו כללי הקדימות של האופרטורים אינם מספיקים.

יש לוודא שאין חלוקה ב-0, כדי להימנע מהודעות שגיאה שיגרמו לעצירת התוכנית.

שאלה 10:

נתון התשבץ הבא:



יש למלא בתשבץ את המלים הבאות:

an, pre, pal , and, edge, leg, eg, all, atom, bang, dal, la, dana, or

הכללים הם:

בכל משבצת יש לרשום אות אחת. המלים נקראות משמאל לימין או מלמעלה למטה. אסור שמלה תיקטע ע"י קו עבה.

כתוב תוכנית פרולוג הפותרת תשבץ זה.

בנוסף לפלט האקטואלי, שרטט (ידנית) את המילוי הנכון לתשבץ בהתאם לפתרון שהתקבל.

הדרכה:

התאם לכל משבצת משתנה חופשי, ולכל מלה רשימה.

אין צורך שהתוכנית תציג פלט נאה, מספיק שהיא תתאים אות לכל משבצת.

נתונה החידה הבאה:

- **ברחוב מסוים ישנה שורה של חמישה בתים,**
- **האנגלי חי בבית האדום,**
- **לספרדי יש כלב,**
- **האיש בבית הירוק שותה קפה,**
- **האוקראיני שותה תה,**
- **הבית הלבן צמוד לבית הירוק,88**
- **האיש שברשותו חתול מעשן מרלבורו,**
- **האיש בבית הצהוב מעשן טיים,**
- **האיש בבית האמצעי שותה חלב,**
- **הנורבגי גר בבית הראשון,**
- **הבעלים של השועל גר בצמוד לאיש שמעשן מונטנה,**
- **מעשן הטיים גר צמוד לאיש שברשותו סוס,**
- **האיש שמעשן לקי-סטרייק שותה מיץ תפוזים,**
- **היפני מעשן פרלמנט,**
- **הנורבגי גר צמוד לבית הכחול.**

<u>מהי ארץ המוצא של הבעלים של הזברה, ומהי ארץ המוצא של האיש שאוהב לשתות מים ?</u>

ההנחות הן: 1. כל אדם מגדל בעל חיים אחד בלבד: זברה או בעל-חיים אחר.

2. כל אדם שותה משקה אחד בלבד: מים או משקה אחר.

כתוב והרץ תכנית בשפת פרולוג המוצאת את פתרון החידה.

יש לייצג באופן מתאים את נתוני החידה כך שהתכנית תוכל למצוא את פתרונה.

שים לב שהנתון הבסיסי הוא בית ולכל בית יש חמישה מאפיינים שונים.

שימו לב לדרך בה "הפעלנו" את המחשב כדי להגיע לתוצאות , זהו בדיוק העניין בתכנות לוגי .

הצעות לשיפורים תיקון טעויות ובכלל :

royrachmany@gmail.co.il