LABORATOR 3:

Ex1. Verificarea unei liste dacă este palindrom

```
rev([ ],[ ]).

rev([H|T],L) :- rev(T,N), append(N,[H],L).

palindrome(L) :- rev(L,L).
```

L = [a,b,a]

palindrome([a,b,a]) :- rev([a,b,a],[a,b,a])

rev([a | b,a], [a,b,a]) :- rev([b,a], N), append(N, [a], [a,b,a])
(N = [a,b]; [a,b]; [a,b,a])

rev([b|a],N) :- rev([a],N2), append(N2,[b],N)
(N2 = [a]; N = [a,b])

rev([a|_], N2) :- rev([ ],N3), append(N3,[a],N2)
(N3 → [ ]; [ ] [a] [a])

rev([ ],N3) ⇒ N3 = [ ]

↳ aici dacă nu ajunge să fie L dă false?

Ex2

\# scrisă de mine

```
remove_duplicates([ ],[ ]).
remove_duplicates([H|T], [H|T2]) :- remove_duplicates(T,T2), not(member(H,T2)).

remove_duplicates([_|T],L) :- remove_duplicates(T,L).
```

\# din laborator

```
remove_duplicates([],[]).
remove_duplicates([H|L],M) :- remove_duplicates(L,M), member(H,M).
remove_duplicates([H|L],[H|M]) :- remove_duplicates(L,M), not(member(H,M)).
```

Ex3. atimes(X, L, N) = True dacă X apare de N ori în L

variantă scrisă de mine care nu face substituțiile adică la good trebuie puse valori

```
atimes(_, [], _).
atimes(X, [H|T], N) :- N >= 0, X is H, N2 is N-1, atimes(X, T, N2).
atimes(X, [_|T], N) :- atimes(X, T, N).
```

variantă laborator care face substituțiile pt. X și N

```
atimes(_, [], 0).
atimes(N, [N|T], X) :- atimes(N, T, Y), X is Y+1.
atimes(N, [H|T], X) :- atimes(N, T, X), H \= N.
```

↳ diferit

```
atimes(3, [3|1,2,1], X) :- atimes(3, [1,2,1], Y)   X is Y+1
atimes(3, [1,2,1], Y)                              X = 0+1 = 1
...
Y=0
```

? `atimes(N, [3|1,2,1], 2) :-`

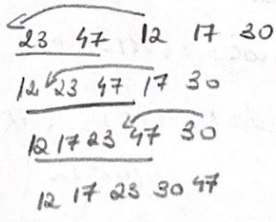
Ex4. Sortarea prin inserție

[47, 23, 12, 17, 30]
H T
[], []

```
insertsort([], []).
insertsort([H|T], L) :- insertsort(T, L1),
                        insert(H, L1, L).

insert(X, [], [X]).
insert(X, [H|T], [X|[H|T]]) :- X < H.
insert(X, [H|T], [H|L]) :- X >= H, insert(X, T, L).
```

[47] 23 12 17 30
considerăm rect. format doar din 47 sortat
23 e înainte



insertsort([47|23,30],L) :- insertsort(T,L1), insert(47,L1,L).
H T

insertsort([23|30],L1) :- insertsort([30],L2), insert(23,L2,L1).
[30] * [30]

insertsort([30|_],L2) :- insertsort([],L3), insert(30,L3,L2).
[30] [] []

insert(30,[],L2) :-
↳ L2 = [30]

* insert(23,[30],L1) :- 23 < 30 => insert(23,[30|_],[23|[30|_]])
L1
[23,30] ?

insert(47,[23,30],L) :- 47 >= 23 => insert(47,[23|30],[23|T2]) :-
[30,47]
insert(47,[30],T2)
[30,47]

insert(47,[30|_],T2) :- 47 >= 30, insert(47,[],T3)
[30|47] ↓
[47]

[23,30,47]

Ex5. QuickSort

quicksort([],[]).
quicksort([H|T],L) :- split(H,T,A,B), quicksort(A,M), quicksort(B,N),
pivotul? append(M,[H|N],L).

[47|23,30]
H T
T2 cele mai mici
cele mai mari
cele mai mici
în N cele mai mari decât H

split(_,[],[],[]).
luăm încă un elem. din tail
split(X,[H|T],[H|A],B) :- H < X, split(X,T,A,B). ← îl punem în A dacă este mai mic
split(X,[H|T],A,[H|B]) :- H >= X, split(X,T,A,B). ← -"- în B -"- mare