Grupo 12:

Bruno Henrique Barbosa Alves,

Guilherme Henrique Fenner Hey,

Gustavo Hammerschmidt,

Pedro Henrique da Silva Churata.

Lista 6 (Listas em Prolog)

PUC - PR

```
/*

Grupo 12:
        Gustavo Hammerschmidt,
        Guilherme Henrique Fenner Hey,
        Bruno Henrique Barbosa Alves,
        Pedro Henrique da Silva Churata.

Lista 06:
*/


% Função auxiliar:  tamanho do vetor.
tamanho_v([],0).
tamanho_v([_|Y], X) :- tamanho_v(Y,W), X is W+1.

% Função auxiliar:  se elemento pertence ao vetor.
pertence([], _, false) :- !.
pertence([X], X, true) :- !.
pertence([X], N, false) :- X=\=N,!.
pertence([X|Y], N, Z) :- X =\=N, pertence(Y,N,Z),!.
pertence([X|_], N,Z) :- X =:= N, Z = true.
membro(S,L,M) :- pertence(L,S,M).

% Questão 1. Maiores([1,2,3,4,5,6,7],5,N) -> N = [6,7]
maior([],[],_).
maior([X|Y],[X|Z],N) :- X > N, maior(Y,Z,N),!.
maior([_|Y],Z,N) :- maior(Y,Z,N).
quest_1(Num,Lista,Z) :-
   nl,write('Elementos maiores do que '),
   write(Num),write(' na lista '),write(Lista),
```

```
   write(' : '), maior(Lista, Z, Num),nl.

% Questão 1: Fim.

% Questão 2. Média lista.
somar([],0).
somar([X],X) :- !.
somar([X|Y],S) :- somar(Y, K), S is X + K,!.

soma(Lista,Soma) :-  somar(Lista, Soma).
size(X,Y) :- tamanho_v(X,Y).

media(Lista, Soma) :-
   soma(Lista,SomaL),size(Lista,Tamanho),
   Tamanho =\= 0, Soma is SomaL/Tamanho, !.
media(_,0).
quest_2(L,S) :-
   nl,write('Média dos elementos na lista '),
   write(L),write(' : '),
   media(L,S),nl.

% Questão 2: Fim.

% Questão 3. Maior que média aritmética.
maiores_media(Lista,Z) :- media(Lista,Media), maior(Lista,Z,Media).
quest_3(L,Z) :-
   nl,write('Elementos maiores do que a média da lista ('),
   media(L,Aux), write(Aux),write(') na lista '),write(L),
   write(' : '), maiores_media(L,Z),nl.

% Questão 3: Fim.
```

```
% Questão 4. Elementos anteriores à pos. n.
ante([],[],_,_).
ante([X|Y],[X|Z],N,Count) :- Count < N, ante(Y,Z,N,Count+1), !.
ante([_|Y],Z,N,Count) :- ante(Y,Z,N,Count+1).
anterior(Num, Lista, Z) :- ante(Lista, Z, Num,0).
quest_4(N,L,X) :-
   nl,write('Elementos anteriores à posição '),write(N),
   write(' na lista '),write(L),
   write(' : '), anterior(N,L,X),nl.

% Questão 4: Fim.

% Questão 5. Elementos posteriores à pos. n. incluindo n
post([],[],_,_).
post([X|Y],[X|Z],N,Count) :- Count >= N, post(Y,Z,N,Count+1), !.
post([_|Y],Z,N,Count) :- post(Y,Z,N,Count+1).
posterior(Num, Lista, Z) :-  post(Lista,Z,Num,0).
quest_5(N,L,X) :-
   nl,write('Elementos posteriores à posição '),write(N),
   write(' na lista '),write(L),
   write(' : '), posterior(N,L,X),nl.

% Questão 5: Fim.

% Questão 6. Lista 1 a n.
ls_1_n(0, []) :- !.
ls_1_n(X,[X|L]) :- X1 is X-1, ls_1_n(X1, L),!.
list1_n(N,[]) :- 1 > N,!.       % reverte <- func. 12.
list1_n(N,Z) :- ls_1_n(N, Aux ), reverte(Aux,Z).
quest_6(N,L) :-
```

```
    nl,write('Lista de 1 a '),
    write(N),write(': '), list1_n(N,L),nl.

% Questão 6: Fim.

% Questão 7. Esquerda e Direita de X.
esqdir([],[],_,_).
esqdir([X|Y],[X|Z],N,Count) :- Count =:= N-1, esqdir(Y,Z,N,Count+1),!.
esqdir([X|Y],[X|Z],N,Count) :- Count =:= N+1, esqdir(Y,Z,N,Count+1),!.
esqdir([_|Y],Z,N,Count) :- esqdir(Y,Z,N,Count+1).
esq_dir(X,Lista,Z) :- esqdir(Lista,Z,X,0).
quest_7(N,L,X) :-
    nl,write('Elementos à esquerda e à direita da posição '),
    write(N),write(' na lista '),
    write(L),write(' : '),
    esq_dir(N,L,X),nl.

% Questão 7: Fim.

% Questão 8. Duas listas.
esq([],[],_,_).
esq([X|Y],[X|Z],N,Count) :- Count < N, esq(Y,Z,N,Count+1),!.
esq([_|Y],Z,N,Count) :- esq(Y,Z,N,Count+1).
dir([],[],_,_).
dir([X|Y],[X|Z],N,Count) :- Count >= N, dir(Y,Z,N,Count+1),!.
dir([_|Y],Z,N,Count) :- dir(Y,Z,N,Count+1).
divide(X,Lista,A,B) :- esq(Lista,A,X,0), dir(Lista,B,X,0).
quest_8(X,L,A,B) :-
    nl,write('Divide a lista '), write(L),write(' em duas na posição: '),
    write(X),write('. '), divide(X,L,A,B),nl.
```

```
% Questão 8: Fim.


% Questão 9. Intervalo entre A e B.
ls_x_y(X, N,[]) :- X-1 =:= N,!.
ls_x_y(X, N, [N|L]) :- N1 is N - 1, ls_x_y(X,N1,L).
intervalo(A,B,[]) :- A > B,!.          % reverte <- func. 12.
intervalo(A,B,Lista):- ls_x_y(A,B,Aux), reverte(Aux,Lista).
quest_9(A,B,L) :-
   nl,write('Lista de '),write(A),write(' a '),
   write(B),write(': '),intervalo(A,B,L),nl.

% Questão 9: Fim.

% Questão 10. Tamanho elementos.
tam([],_).
tam([Y],[A]) :- tamanho_v(Y,A),!.
tam([X|Y], [A|B]) :- tamanho_v(X,A), tam(Y,B),!.
tamanho(List,N) :- tam(List,N).
quest_10(L,N) :-
   nl,write('Tamanho dos elementos da lista '),
   write(L),write(' : '),
   tamanho(L,N),nl.

% Questão 10: Fim.


% Questão 11. Merge listas.
min([],X,X).
min([H|T],M,X) :- H =< M, min(T,H,X).
min([H|T],M,X) :- M < H, min(T,M,X).
minimo([H|T],X) :- min(T,H,X).
```

```
cresc([], []).
cresc([X], [X]) :- !.
cresc([X|Y],[Aux|Z]) :-
   minimo([X|Y], Aux),remover_x_lista_15(Aux,[X|Y],W),cresc(W,Z),!.
ordenar_lista(L,S) :- cresc(L,S).

merg([],X,X):-!.
merg(X,[],X):-!.
merg([X|Y],[W|Z],[X|L]):-  X=<W, merg(Y,[W|Z],L),!.
merg([X|Y],[W|Z],[W|L]):-  W=<X, merg([X|Y],Z,L).

merging(L,L2, M) :-  ordenar_lista(L,O), ordenar_lista(L2,O2), merg(O,O2,M).
quest_11(L,L2,List) :-
   nl,write('Merge das listas '),write(L),
   write(' e '),write(L2),write(' : '),
   merging(L,L2,List),nl.

% Questão 11: Fim.



% Questão 12. Inverte lista.
inverter([],Z,Z).
inverter([X|Y],Z,Acc) :- inverter(Y,Z,[X|Acc]).
reverte(Lista,Y) :- inverter(Lista,Y,[]).
quest_12(X,Y) :-
   nl,write('Inverte a lista '),
   write(X),write(' : '),
   reverte(X,Y),nl.

% Questão 12: Fim.
```

```
% Questão 13. A é sub-lista de B?
sub([],_).
sub([Y],A) :-  membro(Y,A,true).
sub([X|Y],A) :-  membro(X,A,true),  sub(Y,A),!.
sub_lista(Lista, Lista2) :- sub(Lista, Lista2).
quest_13(X,Y) :-
   nl,write(X),write(' é sub-lista de '),
   write(Y),write(' ? '),
   sub_lista(X,Y),nl.

% Questão 13: Fim.

% Questão 14.  Se a Lista está contida em ordem em Lista 2.
co([],[]) :- !.              % co -> contida e em ordem.
co([X|L],[X|S]) :- co(L,S), !.
co(L, [_|S]) :- co(L,S), !.
cont_ordem(List,List2) :- co(List, List2).
quest_14(X,Y) :-
   nl,write(X),write(' está contido em ordem na lista'),
   write(Y),write(' ? '), cont_ordem(X,Y), nl.

% Questão 14: Fim.

% Questão 15. Remove o primeiro X da Lista.
rem_st(_,[],[],_) :- !.     % vazia -> vazia.
rem_st(N,[N],[N],1) :- !.   % mantém último quando igual.
rem_st(N,[B],[B],_) :- N =\= B,!.   % n=2 [1] -> x[1].
rem_st(N,[A|B],Z,Key) :- A =:= N, Key =:= 0, !, rem_st(N,B,Z,1).
rem_st(N,[A|B],[A|Z],Key) :- rem_st(N,B,Z,Key).   % rem_st -> remove first.
```

```
remover_x_lista_15(X,Lista,Z) :- rem_st(X,Lista,Z,0).
quest_15(X,L,Z) :-
   nl,write('Remover a primeira ocorrência de '),write(X),
   write(' da lista '),write(L),write(' : '),
   remover_x_lista_15(X, L, Z),nl.

% Questão 15: Fim.

% Questão 16.  Remove todos os Xs da Lista.
rem(_,[],[]) :- !.    % vazia -> vazia.
rem(N,[N],[]) :- !.   % n=1 [1] -> X = [].
rem(N,[B],[B]) :- N =\= B,!. % n=2 [1] -> x[1].
rem(N,[A|B],Z) :- A =:= N, !,rem(N,B,Z).
rem(N,[A|B],[A|Z]) :- rem(N,B,Z).
remover_x_lista_16(X,Lista,Z) :- rem(X,Lista,Z).
quest_16(X,L,Z) :-
   nl,write('Remover todos os '),write(X),
   write('s da lista '),write(L),write(' : '),
   remover_x_lista_16(X, L, Z),nl.

% Questão 16: Fim.

% Questão 17. Remove elemento na posição n.
rem_p([],[],_,_).
rem_p([X|Y],[X|Z],N,Count) :- Count =\=N, rem_p(Y,Z,N,Count+1),!.
rem_p([_|Y],Z,N,Count) :- rem_p(Y,Z,N,Count+1).
remove_posicao(Posicao,Lista,Z) :- rem_p(Lista,Z,Posicao,0).
quest_17(P,L,X) :-
   nl,write('Remover elemento na posição '),
   write(P),write(' da lista '),write(L),
   write(' : '), remove_posicao(P,L,X),nl.
```

```
% Questão 17: Fim.

% Questão 18. Lista: Y/N.
list18([]).
list18([_]) :-  !.
list18([_|T]) :- list18(T).
lista(X) :- list18(X).
quest_18(X) :-
   nl,write(X),write(' é lista ?'),
   lista(X),nl.

% Questão 18: Fim.

% Questão 19. Nivela a lista.
concatenar([],X,X):-!.
concatenar([X|Y],Z,[X|W]):- concatenar(Y,Z,W).

flat([],[]).
flat([X|Y],Z) :- flat(X,A), flat(Y,B), concatenar(A,B,Z).
flat([X|Y], [X|Z]) :- X \= [], X \= [_|_], flat(Y,Z).

nivela(Lista, Z) :-  flat(Lista, Z),!.
quest_19(L, Z) :-
   nl,write('Nivelar lista '),write(L),
   write(' : '), nivela(L,Z),nl.

% Questão 19: Fim.

% Questão 20. Interseção de duas listas.
inter([],[],_).
```

```
inter([X|Y],[X|Z],L) :-  membro(X,L,true),  inter(Y,Z,L),!.

inter([_|Y],Z,L) :-  inter(Y,Z,L).

intersektion(Lista,Lista2,Inter) :-  inter(Lista,Inter,Lista2).

quest_20(L,L2,In) :-
   nl,write('Interseção das Listas '),write(L),
   write(' e '),write(L2),write(' : '),
   intersektion(L,L2,In),nl.

% Questão 20: Fim.

% Questão 21. Retorna lista com os elementos especificados.

search([],[],_,_).

search([X],[X],N,Count) :- Count =:= N,!.

search([_],[],N,Count) :- Count =\= N.

search([_|Y],Z,N,Count) :- Count =\= N, search(Y,Z,N,Count+1),!.

search([X|Y],[X|Z],N,Count) :- Count =:= N, search(Y,Z,N,Count+1).

list_s([],[],_).

list_s([X],[Z],N) :- search(X,Z,N,0),!.

list_s([X|Y],[R|Z],N) :- search(X,R,N,0), list_s(Y,Z,N),!.

encontra_elementos(Lista, Num, Z) :-  list_s(Lista,Y, Num), nivela(Y,Z).

quest_21(L,N,X) :-
   nl,write('Elementos especificados de posição '),write(N),
   write(' na lista '),write(L),
   write(' : '), encontra_elementos(L,N,X),nl.

% Questão 21: Fim.

% Questão 22. Separar lista em blocos.

block([],[]).

block([X],[[X]]) :- !.

block([X, Y|Z],[ [X]|R ]) :- X \= Y, block([Y|Z], R),!.
```

```
block([X, X|Y],[[X|Z]|T]) :- block([X|Y], [Z|T]),!.

bloco(List, Z) :- block(List, Z).

quest_22(L,Z) :-
   nl,write('Separar lista '),write(L),
   write(' em blocos: '), bloco(L,Z),nl.

% Questão 22: Fim.



% Questão 23. Codificar os blocos da lista dada.

code([],[]).

code([[X|Y]|Z],[[X,N]|W]) :- tamanho_v([X|Y],N), code(Z,W).

code_23(Lista1,Lista2) :- bloco(Lista1, L), code(L, Lista2).

quest_23(L, Z) :-
   nl,write('Codificar lista '),write(L),
   write(' : '), code_23(L, Z),nl.

% Questão 23: Fim.

% Questão 24. Decodifica a lista.

list_24([],[]).

list_24([X],[R]) :-  list_s(X,Z,0),list_s(X,W,1),rep([Z],W,R,W),!.

list_24([[X|Y]|Z],[[R]|W]) :-  rep([X],Y,R,Y), list_24(Z,W),!.

decodifica(Lista, Z) :-  list_24(Lista,B),!, nivela(B,Z).

quest_24(L,Z) :-
   nl,write('Decodificar lista '),write(L),
   write(' : '), decodifica(L, Z),nl.

% Questão 24: Fim.

% Questão 25. Replique cada elemento N vezes.
```

```
rep([],_,[],_).    %fim lista

rep([_|Y],N,Z,0) :-  rep(Y,N,Z,N).  %fim repetição elem. X; vai para Y

rep([X|Y],N,[X|Z],K) :- K > 0, K1 is K - 1,  rep([X|Y],N,Z,K1).

replica(Lista,Num,Z) :-  rep(Lista,Num,Z,Num),!.

quest_25(L,N,X) :-
   nl,write('Replica '),write(N),
   write(' vezes os elementos da lista '),write(L),
   write(' : '), replica(L,N,X),nl.


% Questão 25: Fim.
```