Nome: David Pessoa
RA: 10402647

```
% --------------------------- Exercício 1 ----------------------
onde(_, [], _, -1).
onde(Elem, [X | Lista], Count, Pos) :-
   Count2 is Count + 1,
   (Elem =:= X, Pos is Count2;
   onde(Elem, Lista, Count2, Pos1), Pos is Pos1).

% --------------------------- Exercício 2 ----------------------
ateh(E, [E | L], [E]).
ateh(E, [], []).
ateh(E, [X | L], R) :- ateh(E, L, R1), R = [X | R1].

%--------------------------- Exercício 3 ----------------------
apos(E, [], []).
apos(E, [E | L], L).
apos(E, [X | L], R) :- apos(E, L, R1), R = R1.

%--------------------------- Exercício 4 ----------------------
npri(0, []).
npri(N, Lista) :- N1 is N - 1, npri(N1, Lista1), append(Lista1, [N], Lista).

%--------------------------- Exercício 5 -------------------------
gera_m_mult(_, 0, []).
gera_m_mult(N, M, Lista) :- M1 is M - 1,
   (M mod N =:= 0, gera_m_mult(N, M1, Lista1), append(Lista1, [M], Lista);
   gera_m_mult(N, M1, Lista)).

%--------------------------- Exercício 6 -------------------------
conta(_, [], 0).
conta(N, [X | Lista], F) :- N =:= X, conta(N, Lista, F2), F is F2 + 1.


conta(N, [X | Lista], F) :- N \= X, conta(N, Lista, F).

find_max([], _, MaxF, MaxN, MaxN).
find_max([X | Lista], L, MaxF, MaxN, R) :-
   conta(X, L, F),
   F > MaxF,
   find_max(Lista, L, F, X, R).
find_max([X | Lista], L, MaxF, MaxN, R) :-
   conta(X, L, F),
   F =< MaxF,
   find_max(Lista, L, MaxF, MaxN, R).
```

```
menu :-
  Lista = [3, 3, 4, 3, 4, 8, 8, 8, 8, 8, 3, 3, 3],
  find_max(Lista, Lista, 0, 0, R),
  write("Número com maior frequência na lista: "),
  write(R).
%----------------------- Exercício 7 ----------------------------
mtam([], []).
mtam([X | ListaA], [Y | ListaB]) :- mtam(ListaA, ListaB).

%----------------------- Exercício 8 ----------------------------
tri([], []).
tri([X | Lista], [X, X, X | ListaTriplicada]) :- tri(Lista, ListaTriplicada).

%----------------------- Exercício 9 ----------------------------
subs(_, _, nil, nil). %encontra nó folha

subs(A, B, tree(NO, ND, NE), tree(NR, NDR1, NER1)) :-
  ( NO =:= A -> NR = B ; NR = NO ),
  subs(A, B, ND, NDR1),
  subs(A, B, NE, NER1).

try :-
  Tree = tree(1, tree(2, nil, nil), tree(3, nil, nil)),
  subs(3, 2, Tree, Result), %substitui 3 por 2 na árvore
  writeln(Result).

%----------------------- Exercício 10 ----------------------------
pre(nil, []). %pré-ordem, retorna lista com o percurso feito
pre(tree(NO, FE, FD), Lista) :-
  pre(FE, ListaFE), pre(FD, ListaFD), append([NO], ListaFE, ListaTemp),
append(ListaTemp, ListaFD, Lista).

pos(nil, []). %pós-ordem, retorna lista com o percurso feito
pos(tree(NO, FE, FD), Lista) :-
  pos(FE, ListaFE), pos(FD, ListaFD), append(ListaFE, ListaFD, ListaTemp),
append(ListaTemp, [NO], Lista).

confere([], []). %confere se os percursos são iguais
confere([X | Pre], [Y | Pos]) :- X =:= Y, confere(Pre, Pos).

exec10 :-
  T1 = tree(1, tree(2, nil, nil), tree(3, nil, nil)),
  T2 = tree(3, tree(1, nil, nil), tree(2, nil, nil)),
  pre(T1, R1),
  pos(T2, R2),
confere(R1, R2).
```