**Prolog**

length2([],0).

length2([\_|T], L):-

length2(t,L1), L is L1+1.

Twice( [], []).

twitce ( [ H|T], [H1|T1]) :-

H1 is H\*2 twice (T, T1)

member (X, [X|\_]).//( é verdade que x pertece a uma lista em que ele é o primeiro elemento)

member (X, [\_| T]) :-

member(X,T).

**em haskel**

r(X,Y) :- X==Y

**em Prolog , posso simplesmente fazer:**

r(X,X).

Unificação

Variável livre → uma varíavel que não recebeu um valor

unificação → não existe nenhuma linguagem que tem esse conceito

sum1([],0).

sum1([H|T]) :-

sum1(T,S1), S is S1+H.

count(\_,[],0).

count(X,[X|T],C) :- count(X,T,C1),

C is C1+1.

count(X,[\_|T],C) :- count(X,T,C).

odd(1).

odd(3).

odd(5).

odd(7).

odd(9).

all\_odd([]).

all\_odd([H|T]) :- odd(H), all\_odd(T).

**Acess to second element :**

**functiom [X,Y|T]**

poeta(arnaldo\_antunes).

poeta(castro\_alves).

poeta(drummond).

poeta(joao\_cabral).

poeta(mano\_brown).

morto(drummond).

morto(joao\_cabral).

morto(castro\_alves).

morto(d\_pedro\_I).

poeta\_morto(X):- poeta(X), morto(X).

some\_dead\_poet([H|\_]) :- poeta\_morto(H).

some\_dead\_poet([\_|T]) :- some\_dead\_poet(T).

Second\_part

take(0,\_, []).

take(X,[H|T],[H|T1]) :- Y is X-1, take(Y, T, T1).

Take(3, [1,10,100,100], [1,10,100]).

drop(0, L, L).

drop(X,[\_|T],T1) :- Y is X-1, drop(Y, T, T1).

Take(3, [1,10,100,1000], [10,100,1000])

alternate([],[],[]).

alternate([H1|T1],[H2|T2],[H1,H2|T3]):- alternate(T1,T2,T3).

max([X],X).

max([H|T],H) :- max(T,M), H > M.

max([H|T],M) :- max(T,M), H =< M.

append2([],L,L).

append2([H|T], L ,[H|L1]) :- append2(T,L,L1).

(CONCATENA 2 LISTAS)