TRABALHO DE PROLOG

ALUNOS: Adriano Robson, Eduardo, Vinicius Garcia

1)Pouco se sabe da história passada da família Pinheiro. Existem alguns registos antigos que indicam que o casal José e Maria criou dois filhos, o João e a Ana. Ana teve duas filhas, a Helena e a Joana, também parece ser verdade, segundo os mesmos registos.

Além disso, o Mário é filho do João, pois muito se orgulha ele disso. Estranho também, foi constatar que o Carlos nasceu da relação entre a Helena, muito formosa, e o Mário.

a)Utilizando o predicado progenitor(X,Y) (ou seja, X é progenitor de Y),represente em Prolog todos os progenitores da família Pinheiro.

b)Represente em Prolog as relações: sexo (masculino ou feminino), irmã,irmão, descendente, mãe, pai, avô, tio, primo1

.c)Formule em Prolog as seguintes questões:

1. **O João é filho do José?**
2. **Quem são os filhos da Maria?**
3. **Quem são os primos do Mário?**
4. **Quantos sobrinhos/sobrinhas com um Tio existem na família Pinheiro?**
5. **Quem são os ascendentes do Carlos?**
6. **A Helena tem irmãos? E irmãs?**

```
casal(josé,maria).
casal(mario,helena).
progenitor(ana,helena).
progenitor(ana,joana).
progenitor(josé,ana).
progenitor(josé,joão).
progenitor(joão,mario).
progenitor(mario,carlos).
progenitor(X,Y):-
        casal(Z,X),
        progenitor(Z,Y).
masculino(josé).
masculino(mario).
masculino(joão).
masculino(carlos).
feminino(X):-
        not(masculino(X)).
irmão(X,Y):-
        masculino(X),
        progenitor(Z,X),
        progenitor(Z,Y),
        X \== Y.
irma(X,Y):-
        feminino(X),
        progenitor(Z,X),
```

```
        progenitor(Z,Y),
        X \== Y.
descendente(X,Y):-
        progenitor(X,Y).
descendente(X,Y):-
        progenitor(X,Z),
        descendente(Z,Y).
pai(X):-
        masculino(X),
        progenitor(X,_).
mae(X):-
        feminino(X),
        progenitor(X,_).
avô(X):-
        masculino(X),
        progenitor(X,Y),
        progenitor(Y,_).
tio(X,Z):-
        masculino(X),
        irmão(X,Y),
        X \== Y,
        progenitor(Y,Z).
primo(X,Y):-
        masculino(X),
        progenitor(Z,X),
        progenitor(W,Y),
        irmão(Z,W).
primo(X,Y):-
        masculino(X),
        progenitor(Z,X),
        progenitor(W,Y),
        irma(Z,W).
```

?- progenitor(josé, joão).
true

?- progenitor(maria, X).
X = ana
X = joão

?- primo(mario, X).
X = helena
X = helena
X = joana
X = joana

?- tio(_,Z).
Z = helena

Z = joana

?- descendente(X, carlos).
X = mario
X = helena
X = ana
X = josé
X = josé
X = joão
X = maria
X = maria

?- irmão(X, helena).
false

?- irma(helena, X).
X = joana

2) Implemente as seguintes regras para listas em Prolog.
a)adiciona(X,L1,L2) –onde L2 é a lista que contém o elemento X e a lista L1.
b)apaga(X,L1,L2) –onde L2 é a lista L1 sem o elemento X.
c) membro(X,L) –que é verdadeiro se X pertencer à lista L.
d)concatena(L1,L2,L3) –onde L3 é resultado da junção das listas L2 e L1.
e)comprimento(X,L) –onde X é o número de elementos da lista L.
f)máximo(X,L) –onde X é o valor máximo da lista L (assumir que L contém somente números).
g)media(X,L) –onde X é o valor médio da lista L (assumir que L contém somente números.
h) ordenada(L) | Diz se L está ordenada (ascendentemente).

```
adiciona(X,Y,[X|Y]).
apaga(_,[],Z):-
   write(Z).
apaga(X,[X1|Y],Z):-
   X \== X1,
   apaga(X,Y,[X1|Z]).
apaga(X,[X1|Y],Z):-
   X =:= X1,
   apaga(X,Y,Z).
membro(_,[]):-!.
membro(X,[X1|Y]):-
   X \== X1,
   membro(X,Y).
membro(X,[X1|_]):-
   X=:=X1,
   write("existe este membro").
```

```
concate([], Lista, Lista) :-
        write(Lista),!.
concate([E|Lista1], Lista2, [E|Lista3]) :-
concate(Lista1, Lista2, Lista3).
tam([], 0) :- !.
tam([_|Cauda], N) :-
tam(Cauda, N1),
N is N1 + 1.
max([X], X) :- !.
max([X|Y], X) :-
   max(Y, R),
        (X > R).
max([X|Y], R) :-
   max(Y, R),
        (X =< R).
somarLista([],0).
somarLista([C|L],S) :-
        somarLista(L,A), (S is A+C).

media(X, R2):-
        tam(X, R),
        somarLista(X, R1),
        R2 is R1/R.
```

?- adiciona(5, [1, 2, 3], X).
X = [5, 1, 2, 3]

?- membro(3, [1, 2, 3, 4, 5]).
existe este membro
true

?- concate([1, 2, 3], [4, 5, 6], X).
[4, 5, 6]
X = [1, 2, 3, 4, 5, 6]

?- tam([1, 2, 3, 4, 5], X).
X = 5

?- max([1, 6, 2, 4, 3], X).
X = 6

?- media([1, 2, 3, 4, 5], X).
X = 3

?- apaga(2, [1, 2, 3, 2, 4], L2).
L2 = [1, 3, 4]

3) Usando a tabela d(0,zero), d(1,um), ..., d(9,nove), defina o predicado txt(D,P) que converte uma lista de dígitos numa lista de palavras.Por exemplo, txt([7,2,1],P) resulta em P=[sete,dois,um].

```
d(0, zero).
d(1, um).
d(2, dois).
d(3, tres).
d(4, quatro).
d(5, cinco).
d(6, seis).
d(7, sete).
d(8, oito).
d(9, nove).
txt([], []).
txt([X|Y], Z) :-
        d(X, P),
        txt(Y, N),
        Z = [P | N].
```

?- txt([4, 0, 0, 2], Z).
Z = [quatro, zero, zero, dois]

4)Crie um programa em Prolog que leia um número e calcule e imprima o quadrado desse número, o programa deve continuar a execução até que o usuário digita a palavra ‘stop’.

```
calculadora:-
   write('Digite um numero: '),
   read(X),
        (
         X = 'stop' ->
        nl,
        write('Encerrando a calculadora.');
        number(X) ->
        X1 is X*X,
        nl,
        write('O quadrado de '),
        write(X),
        write(' é '),
        write(X1),
        nl,
        nl,
        calculadora;
        write('Entrada inválida! Por favor, digite um número válido ou "stop" para encerrar.'),
        nl,
        nl,
```

calculadora
).

5) Construa uma base de conhecimento considerando o mapa abaixo. Considere que as setas são o caminho de uma cidade para outra e as letras são a identificação destes caminhos. Considere ainda que cada caminho tem um custo associado a ele conforme indica a tabela abaixo.

Caminho	Custo
A	150
B	90
C	211
D	300
E	50
F	89
G	187
H	254
I	621
J	300
K	41
L	99
M	148
N	163
O	69
P	10
Q	364
R	79
S	193
T	311
U	577
V	150
X	100

Pede-se:

a) Usando o predicado estrada(Identificador,Origem,Destino, Custo), crie um programa para representar esse mapa.

b) Defina o predicado rota(A,B,R,C), que determina todas rotasR que leva da cidade A até a cidade B com o custo C.

c) Defina um predicado rota C(R, A, C) que determina todas rotasR que chegam a cidade A e seus respectivos custos C.

d) Defina um predicado rota S(R, A, C) que determina todas rotas R que saem da cidade A e seus respectivos custos.

e) Defina o predicado rotaM(B,R,C), que determina todas rotas R que chegam a cidade B com o custo C menor do que o valor informado na consulta

```
caminho(a, 150).
caminho(b, 90).
caminho(c, 211).
caminho(d, 300).
```

```
caminho(e, 50).
caminho(f, 89).
caminho(g, 187).
caminho(h, 254).
caminho(i, 621).
caminho(j, 300).
caminho(k, 41).
caminho(l, 99).
caminho(m, 148).
caminho(n, 163).
caminho(o, 69).
caminho(p, 10).
caminho(q, 364).
caminho(r, 79).
caminho(s, 193).
caminho(t, 311).
caminho(u, 577).
caminho(v, 150).
caminho(x, 100).
estrada(cidadeA, cidadeB, b, 90).
estrada(cidadeA, cidadeD, x, 100).
estrada(cidadeA, cidadeG, g, 187).
estrada(cidadeB, cidadeA, a, 150).
estrada(cidadeB, cidadeD, e, 50).
estrada(cidadeB, cidadeE, f, 89).
estrada(cidadeB, cidadeC, c, 211).
estrada(cidadeC, cidadeJ, j, 300).
estrada(cidadeD, cidadeH, h, 254).
estrada(cidadeD, cidadeL, l, 99).
estrada(cidadeF, cidadeI, i, 621).
estrada(cidadeG, cidadeA, d, 300).
estrada(cidadeH, cidadeM, m, 148).
estrada(cidadeI, cidadeJ, k, 41).
estrada(cidadeJ, cidadeL, q, 364).
estrada(cidadeL, cidadeO, s, 193).
estrada(cidadeL, cidadeN, n, 163).
estrada(cidadeN, cidadeP, v, 150).
estrada(cidadeP, cidadeN, o, 69).
estrada(cidadeP, cidadeM, n, 163).
estrada(cidadeQ, cidadeP, t, 311).
estrada(cidadeQ, cidadeL, p, 10).
estrada(cidadeQ, cidadeR, u, 577).

rota(A, B, R, C) :-
    rota(A, B, [A], R, C).

rota(A, A, _, [A], 0).
rota(A, B, Visit, [A | R], C) :-
```

```
    estrada(A, X, _, Custo),
    \+ member(X, Visit),
    rota(X, B, [X | Visit], R, RCusto),
    C is Custo + RCusto.

rotaC(A, R, C) :-
    findall((B, Custo), rota(B, A, _, Custo), R),
    sort(2, @=<, R, Sorte),
    member((A, C), Sorte).

rotaS(R, A, C) :-
  findall((Rota, Custo), estrada(A, _, Rota, Custo), R).

rotaM(B, R, C) :-
  findall((Rota, Custo), (estrada(_, B, Rota, Custo), Custo < C), R).
```

?- rota(cidadeA, cidadeB, Rota, Custo).
Custo = 90,
Rota = [cidadeA, cidadeB]

?- rotaC(cidadeA, Rotas, Custos).
Custos = 0,
Rotas = [(cidadeA,0), (cidadeB,150), (cidadeG,300)]

?- rotaS(Rotas, cidadeA, Custos).
Rotas = [(b,90), (x,100), (g,187)]

?- rotaM(cidadeB, Rotas, 200).
Rotas = [(b,90)]