Atividade 01:

Código:

progenitor(maria, rodrigo). /\* Predicado binário \*/

progenitor(rmãol, rodrigo).

Progenitor(rodrigo, julia).

Progenitor(silvia, julia).

Progenitor(rodrigo, rmão).

Progenitor(ana, rmão).

Masculino(rmãol). /\* Predicado unário \*/

masculino(rodrigo).

Masculino(rmão).

Feminino(maria).

Feminino(ana).

Feminino(julia).

Feminino(silvia).

/\* 2) Regras \*/

rmão(X, Y) :- progenitor(Z, X), progenitor(Z, Y), masculino(X), not(X = Y).

irma(X, Y) :- progenitor(Z, X), progenitor(Z, Y), feminino(X), not(X = Y).

pai(X, Y) :- progenitor(X, Y), masculino(X).

mae(X, Y) :- progenitor(X, Y), feminino(X).

filho(X,Y) :- (pai(Y,X);mae(Y,X)), masculino(X), not(X = Y).

avo(X, Z) :- progenitor(X, Y), progenitor(Y, Z), feminino(X).

avo\_m(X, Z) :- progenitor(X, Y), progenitor(Y, Z), masculino(X).

neto(X, Z) :- progenitor(Z, Y), progenitor(Y, X), masculino(X).

neta(X, Z) :- progenitor(Z, Y), progenitor(Y, X), feminino(X).

filho\_maria(X) :- progenitor(maria, X).

filha\_silvia(X) :- progenitor(silvia, X).

filho\_rodrigo(X) :- progenitor(rodrigo, X), masculino(X).

1. Listar todas as mães: write(‘Todos as mães:’), mae(X, \_).
2. Listar todas as filhas: write(‘Todos as Filhas:’), irma(X, \_).
3. Listar todos os filhos: filho(X,\_)
4. Listar as avós: avo(X, \_)
5. Listar os avôs: avo\_m(X, \_)
6. Listar os netos : neto(X, \_).
7. Listar as netas: neta(X, \_).
8. Listar todas as irmãs: irma(X, \_).
9. Listar todos os irmãos: irmão(X, \_).
10. Listar os filhos de Maria: filho\_maria(X).
11. Listar as filhas de Sílvia: filha\_silvia(X).
12. Listar os filhos de rodrigo: filho\_rodrigo(X).

Atividade 02:

interface("ItfPessoa").

interface("ItfAluno").

interface("ItfProfessor").

interface("ItfCoordenador").

classe("ClsPessoa").

classe("ClsAluno").

classe("ClsProfessor").

classe("ClsCoordenador").

implementa("ClsPessoa", "ItfPessoa").

implementa("ClsAluno", "ItfAluno").

implementa("ClsProfessor", "ItfProfessor").

implementa("ClsCoordenador", "ItfCoordenador").

herda("ClsAluno", "ClsPessoa").

herda("ClsProfessor", "ClsPessoa").

herda("ClsCoordenador", "ClsProfessor").

subclasse(X, Y) :- classe(X), herda(X, Y), classe(Y).

contrata(X, Y) :- classe(X), implementa(X, Y), interface(Y).

herda\_de\_pessoa(X) :- herda(X, "ClsPessoa").

herda\_de\_aluno(X) :- herda(X, "ClsAluno").

herda\_de\_professor(X) :- herda(X, "ClsProfessor").

contrata\_pessoa(X) :- implementa(X, "ItfPessoa").

contrata\_professor(X) :- implementa(X, "ItfProfessor").

filhas\_pessoa(X) :- herda(X, "ClsPessoa"), classe(X).

filhas\_professor(X) :- herda(X, "ClsProfessor"), classe(X).

interface\_aluno(X) :- implementa(X, "ItfAluno"), classe(X).

a) Listar todas as classes que herdam da classe Pessoa:

**Resposta:** herda\_de\_pessoa(X) :- herda(X, "ClsPessoa").

b) Listar todas as classes que herdam da classe Aluno.

**Resposta:** herda\_de\_aluno(X) :- herda(X, "ClsAluno").

c) Listar todas as classes que herdam da classe Professor.

**Resposta:** herda\_de\_professor(X) :- herda(X, "ClsProfessor").

d) Listar todas as classes que contratam a interface Pessoa.

**Resposta:** contrata\_pessoa(X) :- implementa(X, "ItfPessoa").

e) Listar todas as classes que contratam a interface Professor.

**Resposta:** contrata\_professor(X) :- implementa(X, "ItfProfessor").

f) Listar todas as classes filhas da classe Pessoa.

**Resposta:** filhas\_pessoa(X) :- herda(X, "ClsPessoa"), classe(X).

g) Listar todas as classes filhas da classe Professor.

**Resposta:** filhas\_professor(X) :- herda(X, "ClsProfessor"), classe(X).

h) Listar a interface que a classe Aluno implementa.

**Resposta:** interface\_aluno(X) :- implementa(X, "ItfAluno"), classe(X).

Atividade 03:

a) Aviação:

Fatos:

aviao(boeing, 747).

aviao(airbus, a380).

companhia\_aerea(emirates, dubai).

Regras:

eh\_grande(Aviao) :- aviao(A, Aviao), Aviao > 500.

voa\_companhia(Companhia, Aviao) :- aviao(A, Aviao), companhia\_aerea(Companhia, C), A == C.

eh\_moderno(Aviao) :- aviao(A, Aviao), A == boeing.

Consultas:

aviao(boeing, X).

eh\_grande(X).

voa\_companhia(emirates, X).

aviao(airbus, Y), Y > 700.

eh\_moderno(X).

b) Comércio:

Fatos:

produto(monitor, dell).

produto(mouse, logitech).

loja(casas\_bahia, sao\_paulo).

Regras:

produto\_popular(Produto) :- produto(P, Produto), P == mouse.

vende\_loja(Loja, Produto) :- produto(P, Produto), loja(Loja, L), P == dell.

produto\_em\_promocao(Produto) :- produto(P, Produto), P == monitor.

Consultas:

produto(mouse, X).

produto\_popular(X).

vende\_loja(casas\_bahia, X).

produto\_em\_promocao(X).

loja(X, sao\_paulo), vende\_loja(X, Y).

c) Medicina:

Fatos:

medicamento(remedio\_1, dor\_de\_cabeca).

medicamento(remedio\_2, gripe).

especialidade\_medica(neurologia, dor\_de\_cabeca).

Regras:

tratamento\_especialidade(Medicamento, Especialidade) :- medicamento(M, Medicamento), especialidade\_medica(Especialidade, E), M == E.

medicamento\_para(Medicamento, Sintoma) :- medicamento(Medicamento, S), S == Sintoma.

eh\_generico(Medicamento) :- medicamento(M, Medicamento), M == remedio\_2.

Consultas:

medicamento(remedio\_1, X).

tratamento\_especialidade(X, neurologia).

medicamento\_para(X, dor\_de\_cabeca).

eh\_generico(X).

especialidade\_medica(X, Y).

d) Meio ambiente:

Fatos:

animal(leao).

animal(tigre).

ecossistema(savana, leao).

Regras:

vive\_ecossistema(Animal, Ecossistema) :- animal(A, Animal), ecossistema(E, Ecossistema), A == E.

animal\_ameacado(Animal) :- animal(A, Animal), A == tigre.

ecossistema\_em\_risco(Ecossistema) :- ecossistema(S, Ecossistema), S == savana.

Consultas:

animal(X).

vive\_ecossistema(X, savana).

animal\_ameacado(X).

ecossistema\_em\_risco(X).

vive\_ecossistema(X, floresta).

e) Tecnologia:

Fatos:

sistema\_operacional(windows).

sistema\_operacional(mac\_os).

linguagem\_programacao(java).

Regras:

popular(Sistema) :- sistema\_operacional(Sistema), Sistema == windows.

linguagem(Sistema, Linguagem) :- sistema\_operacional(Sistema), Linguagem == java.

moderno(Sistema) :- sistema\_operacional(Sistema), Sistema == mac\_os.

Consultas:

sistema\_operacional(X).

popular(X).

linguagem(windows, X).

moderno(X).

linguagem(X, java).