1. Um mapa de um certo jogo é formado por várias salas numeradas. Algumas salas estão conectadas, outras não. Para transitar de uma sala para outra é necessário que haja uma conexão entre as duas salas, neste jogo as conexões são de mão-única, ou seja, se há uma conexão de 1 para 2, é possível ir da sala 1 para a sala 2 mas o contrário não é verdade.

Considere a seguinte lista de conexões:
c(1,2). c(3,4). c(5,6). c(7,8). c(9,10). c(12,13). c(13,14). c(15,16). c(17,18). c(19,20). c(4,1). c(6,3). c(4,7). c(6,11). c(14,9). c(11,15). c(16,12). c(14,17). C(16,19).

Onde c(X,Y) indica que é possível ir diretamente da sala X para a sala Y.

a) Escreve um predicado vai(X,Y) que diz em que locais do mapa é possível chegar encadeando transições nessa base de conhecimento.
b) Em quais salas é possível chegar partindo da sala 1?
c) Há alguma sala inalcançável a partir das demais?
d) Quais salas são becos sem saída?

2. Considere os seguintes predicados com informações sobre viagens:

```
carro(saoPaulo,campinas).
carro(campinas,paulinea).
carro(valmont,saarbruecken).
carro(valmont,metz).

trem(metz,frankfurt).
trem(saarbruecken,frankfurt).
trem(metz,paris).
trem(saarbruecken,paris).

aviao(frankfurt,bangkok).
aviao(frankfurt,singapura).
aviao(paris,losAngeles).
aviao(bangkok,saoPaulo).
aviao(singapura,saoPaulo).
aviao(losAngeles,saoPaulo).
```

carro(X,Y) indica que é possível ir de carro de X para Y. trem(X,Y) indica que é possível ir de trem de X para Y e aviao(X,Y) indica que é possível ir de avião de X para Y.

a) Escreva um predicado viagem(X,Y) que determina se é possível ir de X para Y através de alguma combinação de deslocamentos. Ex: viagem(paris, paulinea) deve retornar True.
b) Saber que é possível viajar é uma informação interessante, mas melhor ainda seria saber qual sequência de viagens devem ser feitas. Escreva um predicado viagem(X,Y,Z) que diz a rota que deve ser feita. Por exemplo, para a consulta viagem(paris, paulinea, Z) o programa deve retornar:
Z = vai(paris, losAngeles, vai(losAngeles, saoPaulo, vai(saoPaulo, campinas, vai(campinas, paulinea))))
c) Extenda o predicado definido anteriormente para dizer que tipo de transporte deve ser utilizado nas transições. Para a consulta viagem(paris, paulinea, Z) o programa deve retornar:
Z = vai(paris, losAngeles, aviao, vai(losAngeles, saoPaulo, aviao, vai(saoPaulo, campinas, carro, vai(campinas, paulinea, carro))))

3. O problema das torres de Hanoi pode ser descrito da seguinte forma: Dado um conjunto de N discos de diferentes tamanhos, dispostos em ordem decrescente de tamanho na haste esquerda (com o maior disco em baixo), movê-los para a haste central usando a haste da direita como auxiliar. A movimentação segue as seguintes restrições:
Apenas um disco pode ser movido por vez.
O movimento consiste em tirar um disco do topo de uma das pilhas e colocar no topo de outra pilha.
Um disco maior nunca pode estar acima de um disco menor.
Escreva um predicado que resolve esse problema.

Uma forma recursiva de pensar na solução é: dados N discos, mova N-1 para a haste da direita, mova o disco final para a haste central e depois mova todos os discos da haste direita para a haste central.

Para escrever os movimentos na tela use o seguinte predicado:
move(X,Y) :- write(“mova da haste ”), write(X), write(“ para a haste ”), write(Y), nl.

4. Considere as seguintes informações:
pai(a,b).
pai(a,c).
pai(b,d).
pai(b,e).
pai(c,f).
onde pai(X,Y) indica que X é pai de Y.

a) defina um predicado irmao(X,Y) que é verdade se X e Y são irmãos.
b) defina um predicado primo(X,Y) que é verdade se X e Y são primos.
c) defina um predicado neto(X,Y) que é verdade se X é neto de Y.
d) defina um predicado descendente(X,Y) que é verdade se X é descendente de Y.
e) faça a árvore de busca (algoritmo da resolução) para cada um dos predicados que você criou. Basta expandir até achar a segunda solução (se houver).