Resolução do exame primeira chamada 2001

1)

Para provar que $T \vdash p(a)$ tinhamos que provar por resolução que $T \cup \{\neg p(a)\} \vdash \Box$.

Bastava usar a equivalencia para obter a frase $p(X) \leftarrow (q(X) \ \vee \ r(X))$. Desta frase obtinha-se duas cláusulas: $p(X) \ \vee \ \neg q(X)$ e $p(X) \ \vee \ \neg r(X)$.

Resolvendo a segunda com $\neg p(a)$ obtinhamos $\neg r(a)$.

Resolvendo com $r(a)$ que faz parte de T obtemos $\Box$.

2)

```
rep(nil,_,_,_,nil).
rep(t(X,nil,nil),N,M,M,t(N,nil,nil)):- M < X.
rep(t(X,nil,nil),N,M,X,t(N,nil,nil)):- M >= X.
rep(t(_,E,D),N,M,M1,t(_,E1,D1)):-
             rep(E,N,M,M2,E1),
             rep(D,N,M2,M1,D1).

repmin(Tree,Nova):- rep(Tree,N,0,N,Nova).
```

3)

Uma simples árvore de prova não esquecendo de mostrar todos os ramos (todas as alternativas) e respectivos unificadores mais gerais.

4)

?X = p(Y) dá $X = p(Y)$ e Y sem substituição.

?not X = p(Y) falha porque ?$X = p(Y)$ sucede como vimos na respostas anterior. Assim não há substituições para X e Y.

?$X = p(X)$ não termina pois é uma tentativa de fazer uma unificação circular (substituição infinita de X por $p(X)$).

5)

Basta jogar com o troque da dupla negação para não "estragar" a lista formato. Depois paramos quando não for possivel unificar a lista formato com a lista a imprimir (o que quer dizer que já não há elementos suficientes para imprimir!).

```
forma(F,L):- \+ append(F,_,L).
forma(F,L):-
      \+ \+ (append(F,R,L), write(F)), nl,
      append(T,R,L), \+ \+ (T = F), forma(F,R).
```

6)

Bastava explicar que o facto de o Prolog dar prioridade a explorar a árvore de prova em profundida pode levar à exploração de um ramo infinito. Assim o Prolog nunca conseguia recuperar dessa situação e continuar a exploração de outros ramos. (a justificação devia ser sustentada com exemplos de queries e programas).

7)

```
:- dynamic info/4.

pares(X):-
    faz_pares(X).

faz_pares([X/Y|L]):-
     acha(X,Y),
     retract(info(X,_,_,_)),
     retract(info(Y,_,_,_)),
     faz_pares(L).
faz_pares([]).

acha(Nom,Y):-
  info(Nom,S,Id,La),
  findall(p(abs(Id2-Id),Nom2),
          (info(Nom2,S2,Id2,La),Nom \== Nom2,S \== S2),
          L),
  sort(L,[p(DId,Y)|_]),
  DId < 15.
acha(Nom,Y):-
  info(Nom,S,Id,_),
  info(Y,S2,Id2,_),
  Nom \== Y,
  S \== S2,
  abs(Id - Id2) < 15.
```