1.

a) O algoritmo "subir a colina" é um algoritmo de pesquisa de solução que utiliza unicamente informação local. Na sua versão mais simples, o primeiro sucessor gerado cuja avaliação seja melhor que o pai, é escolhido como sucessor. Na sua versão "subida íngreme", gera sempre todos os sucessores do nivel seguinte, e escolhe o que tiver melhor avaliação. As escolhas são irreversiveis. Isto é, uma vez escolhido um nó, ele pertencerá sempre ao caminho seleccionado.
Versão mais simples: a, b, e, j, …
Versão "subida íngreme": a,b,f, …

b)
```
scolina(Ef,[]):- objectivo(Ef).
scolina(Ea,[Ea|R]):- suc(Ea,ESucs),
    ( ESucs==[], write('insucesso')
    ; select(ESucs,Eseg),
      scolina(Eseg,R)
    ).

suc(Ea, Esuc):-
    findall( F, filho(Ea,F), Esuc).

select([E1],E1).
select([E1,E2|OEs],Eseg):-
    estimativa(E1,C1),
    estimativa(E2,C2),
    ( C1<C2, select([E1|OEs],Eseg)
    ; select([E2|OEs],Eseg)
    ).


inicio(a).
objectivo(x).
filho(a,b).
filho(a,d).
filho(a,c).
filho(b,e).
filho(b,f).
filho(b,g).
filho(c,h).
filho(d,i).
filho(e,j).
estimativa(b,4).
estimativa(c,5).
estimativa(d,6).
estimativa(e,3).
```

2.

a)

Q x adv(X) → rico(X).	**1 adv(eva)**
Qx rico(X) → grande (carro(X))	**2 adv(X) U rico(X)**
Qx grande(carro(X)) → gasta(carro(X))	**3 ~rico(X) U grande(carro(X))**
	4 ~grande(carro(X)) U gasta(carro(X))

b) Resolução:
5 ~gasta(carro(eva))

de 5 e 4 ~grande(carro(eva)) 6
de 6 e 3 ~rico(eva) 7
de 7 e 2 ~adv(eva) 8
de 8 e 1 inconsistência

3.

```
:- dynamic erro/1.

ln:- write('Escreva uma frase: '),nl,
     fread(s,0,0,Frase), transf_lista(Frase,Lista_Pal),
     (  verifica_frase(Lista_Pal,[]) ;
        ( erro(semantico),write('erro semantico')
        ; write('erro sintatico')
        )
     ).

verifica_frase-->
     sintg_interrogativo(Q,N,Sujeito),
     sintg_verbal(N,Sujeito,Accao,Objecto),
     {resposta(Q,Sujeito,Accao,Objecto)}.

verifica_frase -->
     sintg_nominal(_-N,LSujeito),sintg_verbal(N,LSujeito,Accao,Objecto),
     {resposta_afirm(LSujeito,Accao,Objecto)}.


sintg_interrogativo(Q,N,Sujeito)--> pron_int(Q,G-N),
     det(G-N),sintg_nominal_r(G-N,Sujeito),[que].
sintg_interrogativo(Q,N,Sujeito)--> pron_int(Q,G-N),
     sintg_nominal_r(G-N,Sujeito).

sintg_nominal(G-p,[Nome1|[RNome]])--> sintg_nominal1(_,Nome1),[e],
     sintg_nominal1(_,RNome).
sintg_nominal(G-N,[Nome])--> sintg_nominal1(G-N,Nome).

sintg_nominal1(G-N,Sujeito)--> det(G-N),sintg_nominal_r(G-N,Sujeito).
sintg_nominal1(G-N,Sujeito)--> sintg_nominal_r(G-N,Sujeito).
sintg_nominal_r(G-N,(Nome,Adj))--> nome(G-N,Nome), adjectivo(G-N,Adj).
sintg_nominal_r(G-N,(Nome,_))--> nome(G-N,Nome).



sintg_verbal(N,Sujeito,Accao,Objecto)-->
     verbo(N,Sujeito,Accao), complemento(Objecto).

complemento(Obj)--> prep(G-N),nome(G-N,Obj).
complemento(Obj)--> prep(G-N),nome(G-N,_),adjectivo(G-N,Obj).

%%%%%%  respostas
```

```
resposta(Q,Sujeito,Accao,Objecto):-
     respostas_(Sujeito,Accao,Objecto,Lista),
     ( ( Q=ql,write(Lista) )
     ; ( length(Lista,Nlista),write(Nlista) )
    ), nl.

respostas_(Sujeito,Accao,Objecto,Lista):-
     Sujeito=(Nome,Adj),
     var(Adj), Facto=..[Accao,X,Objecto],
     ( setof( X, Facto, Lista)
     ; Lista=[] ).
respostas_(Sujeito,Accao,Objecto,Lista):-
     Sujeito=(Nome,Adj),
     nonvar(Adj), Facto1=..[ser,X,Adj], Facto2=..[Accao,X,Objecto],
     ( setof( X, (Facto1,Facto2), Lista)
     ; Lista=[] ).


resposta_afirm([],_,_):- write(concordo).
resposta_afirm([(Nome1,_)|ONomes],Accao,Objecto):-
     Facto=..[Accao,Nome1,Objecto],
     Facto,
     resposta_afirm(ONomes,Accao,Objecto).
resposta_afirm(_,_,_):- write(discordo).

%%%%%%  gramatica
pron_int(qt,f-p)--> ['Quantas'];[quantas].
pron_int(ql,_-p)--> ['Quais'];[quais].
det(m-s)--> [o].
det(f-s)--> [a].
det(f-p)--> [as].
det(f-s)--> ['A'].
det(m-s)--> ['O'].
prep(m-s)--> [no].
prep(m-s)--> [ao].
adjectivo(m-s,europeu)--> [europeu].
adjectivo(f-p,latino)--> [latinas].
nome(f-p,equipa)--> [equipas].
nome(m-s,campeonato)--> [campeonato].
nome(m-s,portugal)--> ['Portugal'];[portugal].
nome(m-s,brasil)--> ['Brasil']; [brasil].
nome(f-s,franca)--> ['Franca'].
nome(f-s,alemanha)--> ['Alemanha'];[alemanha].
nome(m-s,grupoA)-->[grupoA].
verbo(p,Sujeito,jogar)--> [jogam].
verbo(s,Sujeito,jogar)--> [joga].
verbo(p,Sujeito,pertencer)--> [pertencem],{equipa(Sujeito);assert(erro(semantico)),fail}.
verbo(s,Sujeito,pertencer)--> [pertence],{equipa(Sujeito);assert(erro(semantico)),fail}.

%%%%% base conhecimento
ser(portugal,latino).
ser(espanha,latino).
ser(italia,latino).
```

```
ser(franca,latino).
ser(romenia,latino).
ser(noruega,latino).

pertencer(portugal,grupoA).
pertencer(alemanha,grupoA).
jogar(portugal,europeu).
jogar(alemanha,europeu).
jogar(franca,europeu).

equipa(equipa).
equipa(portugal).
equipa(alemanha).
equipa(brasil).
equipa([]).
equipa([(Nome,_)|R]):- equipa(Nome),equipa(R).
equipa((Nome,_)):- equipa(Nome).


transf_lista(Frase,Lista_Pal):-
      string_chars(Frase,Lista_char),
      faz_palavras(Lista_char,[],Lista_Pal).
faz_palavras([13],L_char1,[Pal1]):-
      atom_chars(Pal1,L_char1).
faz_palavras([32|T],L_char1,[Pal1|T1]):-
      atom_chars(Pal1,L_char1),
      faz_palavras(T,[],T1).
faz_palavras([X|T],L_char,L_Pal):-
      append(L_char,[X],L_char1),
      faz_palavras(T,L_char1,L_Pal).
```

4.

a)

$$
\begin{aligned}
H(\text{nota}) &= \frac{4}{10}\left[-0-\frac{4}{4}\times\log\left(\frac{4}{4}\right)\right] \\
&+ \frac{3}{10}\left[-\frac{2}{3}\times\log\left(\frac{2}{3}\right)-\frac{1}{3}\times\log\left(\frac{1}{3}\right)\right] \\
&+ \frac{3}{10}\left[-\frac{3}{3}\times\log\left(\frac{3}{3}\right)-0\right] \\
&= 0+\frac{3}{10}\left[-\frac{2}{3}\times\log 2+\frac{2}{3}\times\log 3+\frac{1}{3}\times\log 3\right]+0 \\
&= \mathbf{0{,}275}
\end{aligned}
$$

$$\begin{aligned}
H(univ.) &= \frac{4}{10}\left[-\frac{3}{4}\times\log\left(\frac{3}{4}\right)-\frac{1}{4}\times\log\left(\frac{1}{4}\right)\right] \\
&+ \frac{3}{10}\left[-\frac{2}{3}\times\log\left(\frac{2}{3}\right)-\frac{1}{3}\times\log\left(\frac{1}{3}\right)\right] \\
&+ \frac{3}{10}[0] \\
&= \frac{4}{10}\left[-\frac{3}{4}\times\log 3+\frac{3}{4}\times\log 4+\frac{1}{4}\times\log 4\right] \\
&+ \frac{3}{10}\left[-\frac{2}{3}\times 2+\frac{2}{3}\times 3+\frac{1}{3}\times\log 3\right] \\
&= \mathbf{0,6}
\end{aligned}$$

$$\begin{aligned}
H(recom.) &= \frac{4}{10}\left[-\frac{1}{4}\times\log\left(\frac{1}{4}\right)-\frac{3}{4}\times\log\left(\frac{3}{4}\right)\right] \\
&+ \frac{6}{10}\left[-\frac{4}{6}\times\log\left(\frac{4}{6}\right)-\frac{2}{6}\times\log\left(\frac{2}{6}\right)\right] \\
&= \frac{4}{10}\left[\frac{1}{4}\times\log 4-\frac{3}{4}\times\log 3+\frac{3}{4}\times\log 4\right] \\
&+ \frac{3}{5}\left[-\frac{2}{3}\times\log 2+\frac{2}{3}\times\log 3+\frac{1}{3}\times\log 3\right] \\
&= \mathbf{0,875}
\end{aligned}$$

O atributo de menor entropia é <u>nota</u> (raiz da árvore).

<u>Caso 1: nota =14</u>

$$H(univ.) = \frac{1}{4}\times 0+\frac{1}{4}\times 0+\frac{2}{4}\times 0 = \mathbf{0}$$

O atributo universidade possui valor de entropia nulo. É suficiente para a identificação dos estudantes.
Caso 1.1: nota=14 e universidade=top_10 => Não aceite
Caso 1.2: nota=14 e universidade=top_20 => Não aceite
Caso 1.3: nota=14 e universidade=top_30 => Não aceite

<u>Caso 2: nota =15</u>

$$H(univ.) = \frac{1}{3}\times 0+\frac{1}{3}\times 0+\frac{1}{3}\times 0 = \mathbf{0}$$

O atributo universidade possui valor de entropia nulo. É suficiente para a identificação dos estudantes.
Caso 2.1: nota=15 e universidade=top_10 => Aceite
Caso 2.2: nota=15 e universidade=top_20 => Aceite
Caso 2.3: nota=15 e universidade=top_30 => Não aceite

<u>Caso 3: nota =16</u>

$$H(univ.) = \frac{2}{3}\times 0+\frac{1}{3}\times 0 = \mathbf{0}$$

O atributo universidade possui valor de entropia nulo. É suficiente para a identificação dos estudantes.
Caso 3.1: nota=16 e universidade=top_10 => Aceite
Caso 3.2: nota=16 e universidade=top_20 => Aceite

b)

Regra 1:	SE nota=14 E universidade=top_10	ENTÃO	Não aceite
Regra 2:	SE nota=14 E universidade=top_20	ENTÃO	Não aceite
Regra 3:	SE nota=14 E universidade=top_30	ENTÃO	Não aceite
Regra 4:	SE nota=15 E universidade=top_10	ENTÃO	Aceite
Regra 5:	SE nota=15 E universidade=top_20	ENTÃO	Aceite
Regra 6:	SE nota=15 E universidade=top_30	ENTÃO	Não aceite
Regra 7:	SE nota=16 E universidade=top_10	ENTÃO	Aceite
Regra 8:	SE nota=16 E universidade=top_20	ENTÃO	Aceite