COMPITO DI INTELLIGENZA ARTIFICIALE

14 Gennaio 2003 - Tempo a disposizione: 2 ore e 30 min.

Esercizio 1

Si traducano le seguenti frasi nella logica dei predicati del primo ordine, poi in forma a clausole:
1) ciascun bambino che non è orfano ha almeno un genitore
2) i genitori si prendono cura dei propri bambini
2) David è un bambino e nessuno si prende cura di lui.

Si usi poi il principio di risoluzione per dimostrare che David è orfano.

Esercizio 2

Si scriva un predicato Prolog `split(L1,N,L2,L3)` che data la lista `L1` ed un numero intero `N` in ingresso restituisca in uscita due liste, `L2,L3`, la prima contenente i numeri maggiori di `N` e la seconda quelli minori od uguali:

Esempio:

```
?-split([3,1,7,5],4,L1,L2)
```

darà come risultato:

```
L1=[7,5]
L2=[3,1]
```

Esercizio 3

Si consideri il seguente programma Prolog, che calcola il minimo di una lista:

```
min(A,B,A) :- A < B, !.
min(A,B,B).

minlist([X],X):- !.
minlist([H|T],M) :-
     minlist(T,M1),
     min(H,M1,M).
```

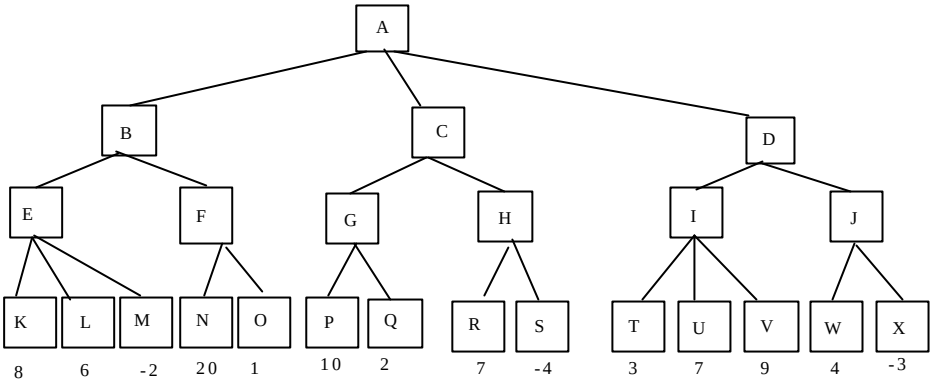
Si rappresenti l'albero SLD corrispondente all'invocazione

```
?- minlist([1,2,3],2).
```

Facoltativo: Il goal precedente ha successo. Il risultato non è corretto: come dovrebbe essere modificato il programma per avere il comportamento corretto?

Esercizio 4:

Si consideri il seguente albero di gioco dove i punteggi sono tutti dal punto di vista del primo giocatore:



Supponendo che il primo giocatore sia MAX, quale mossa dovrebbe scegliere?
Si risolva l'esercizio tramite l'algoritmo MIN-MAX. Successivamente si mostrino i tagli alfa-beta.

SOLUZIONE

Esercizio 1

Trasformazione in clausole:

1 $\forall X$, bimbo(X) **L** ~orfano(X) $\Rightarrow \exists Y$, genitore(Y,X).
~bimbo(X) **V** orfano(X) **V** genitore(f(X),X)

2 $\forall X, \forall Y$, bimbo(X) **L** genitore(Y,X) $\Rightarrow$ ha_cura(Y,X))
~bimbo(X) **V** ~genitore(Y,X) **V** ha_cura(Y,X)

3 bimbo(david) **L** ~ $\exists Y$, ha_cura(Y,david).

3a bimbo(david)

3b ~ha_cura(X,david)

Query negata:

G ~orfano(david)

Risoluzione:

4 (G+1) ~bimbo(david) **V** genitore(f(david),david)

5 (4+2) ~bimbo(david) **V** ha_cura(f(david),david)

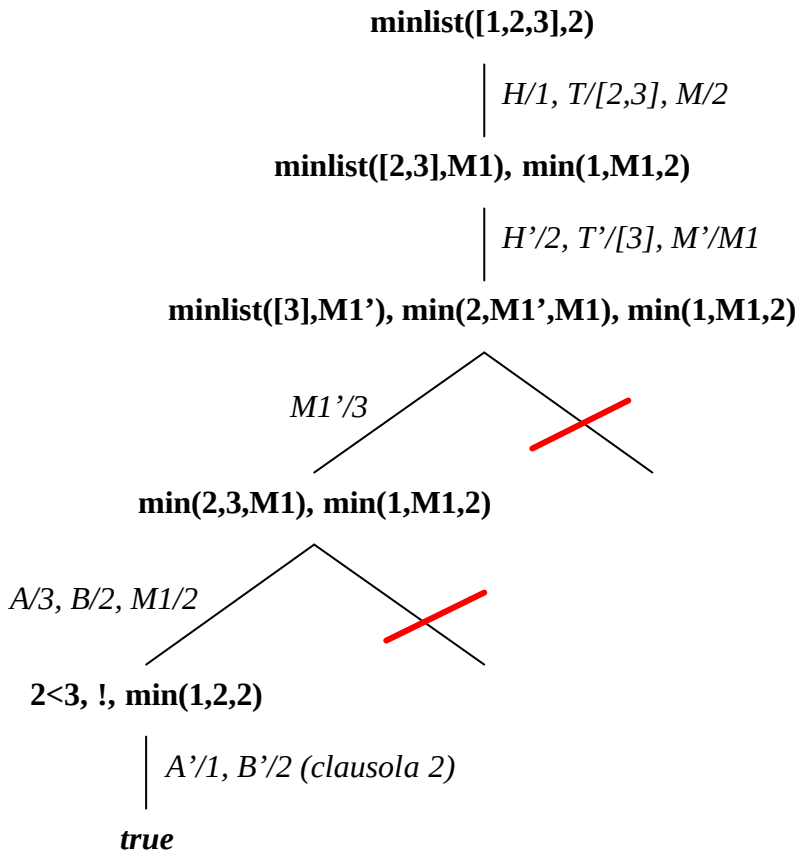
6 (5+3b) ~bimbo(david)

7 (6+3a) []

Esercizio 2

```
split([],Num,[],[] ).
split([A|Rest],Num,[A|L1],L2) :- A>Num,split(Rest,Num,L1,L2).
split([A|Rest],Num,L1,[A|L2]) :- A<=Num,split(Rest,Num,L1,L2).
```

Esercizio 3



Facoltativo:
Il risultato, ovviamente, non è corretto (si dichiara è vero che il minimo della lista [1,2,3] è 2).

Il comportamento è sbagliato a causa del predicato *min/3*. Infatti, il cut nella prima clausola di *min/3* viene effettuato *dopo* l'unificazione del parametro di output. Di conseguenza, visto che l'unificazione col parametro di output fallisce, il cut non viene eseguito ed ha successo la seconda clausola di *min/3*.

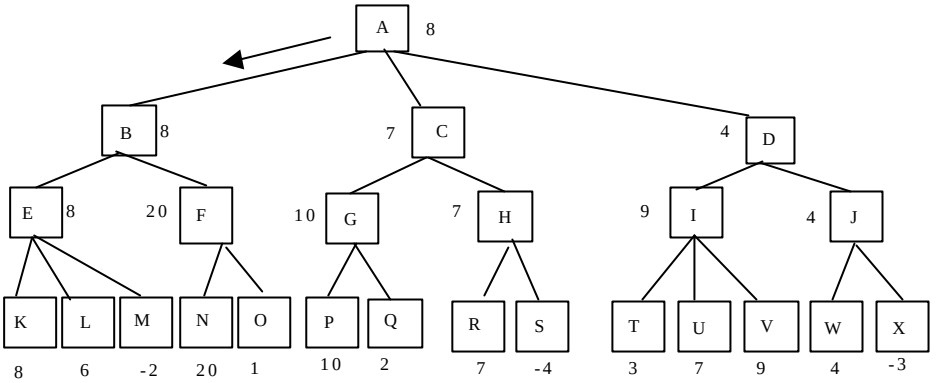
Una possibile soluzione è riscrivere *min/3* in maniera logica, per esempio:

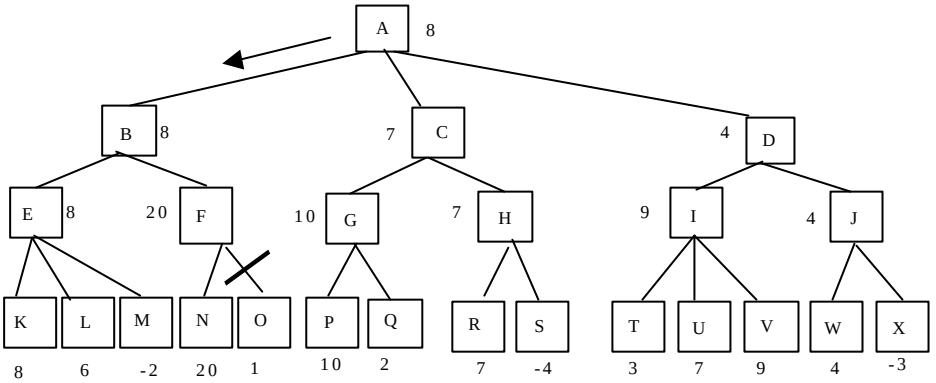
```
min(A,B,A) :- A < B.
min(A,B,B) :- A >= B.
```

oppure si può mettere l'unificazione del parametro di output *dopo* l'esecuzione del cut:

```
min(A,B,M) :- A < B, !, A=M.
min(A,B,B) :- A >= B.
```

Esercizio 4


A 8
B 8
7 C
4 D
E 8
20 F
10 G
7 H
9 I
4 J
K L M N O P Q R S T U V W X
8 6 -2 20 1 10 2 7 -4 3 7 9 4 -3


A 8
B 8
7 C
4 D
E 8
20 F
10 G
7 H
9 I
4 J
K L M N O P Q R S T U V W X
8 6 -2 20 1 10 2 7 -4 3 7 9 4 -3