COMPITO DI INTELLIGENZA ARTIFICIALE (v.o.) – PARTE I
FONDAMENTI DI INTELLIGENZA ARTIFICIALE

14 Gennaio 2005 (Tempo a disposizione 2h; su 32 punti)

Esercizio 1: (punti 7)
Si traducano le seguenti frasi nella logica dei predicati del primo ordine, poi in forma a clausole:

- Esiste almeno un paziente a cui piacciono tutti i medici
- A nessun paziente piace alcun ciarlatano

Si usi poi il principio di risoluzione per dimostrare che nessun medico è un ciarlatano.

Esercizio 2 (punti 8)
Si consideri il seguente programma Prolog:

```
closure([X],[X]):- !.
closure([A<B|T],[A<B|L]):-
    membchk(B<C,T),
    not membchk(A<C,T),!,
    closure ([A<C|T],L).
closure([A<B|T],[A<B|L]):-
    closure(T,L).

membchk(X,[X|_]):- !.
membchk(X,[_|T]):- membchk(X,T).
```

Si disegni l'albero SLDNF relativo al goal:

```
?- closure([a<b,b<c],L).
```

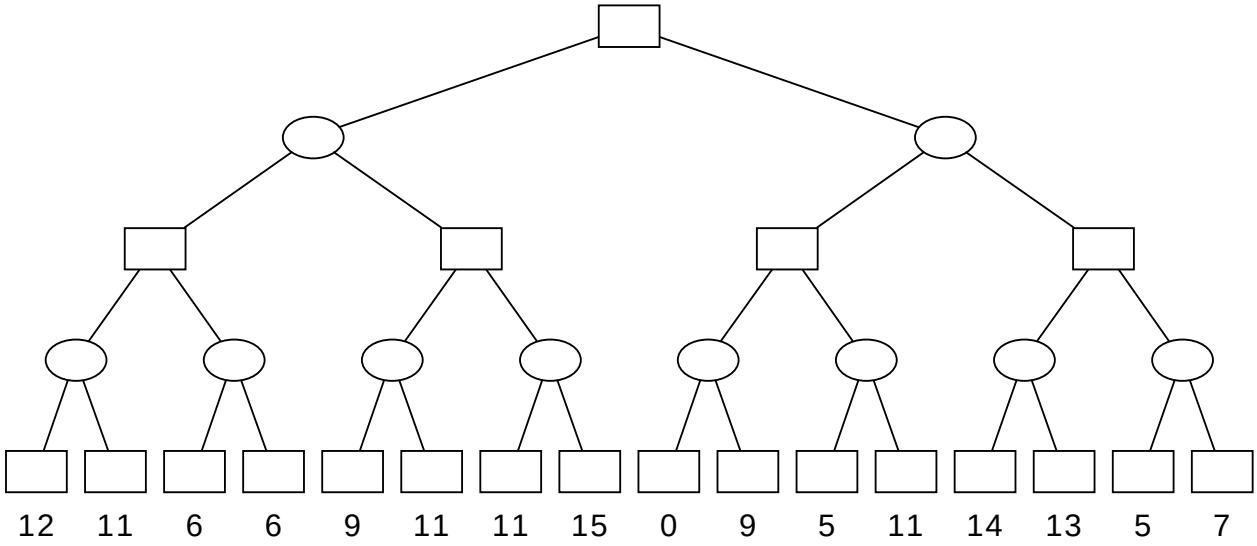
e si dica qual è la risposta calcolata.

Esercizio 3 (punti 6)
Si scriva un programma Prolog *listpart*(*Lin*1,*Lin2, Lout*) che data una lista di elementi interi positivi *Lin*1, produca una lista in uscita *Lout* contenente gli elementi di *Lin2* che corrispondono alle posizioni indicate nella lista *Lin*1 (supposti sempre esistenti). Si scrivano esplicitamente tutti i predicati Prolog usati nella soluzione.

Esempi:
?-listpart([1,4,3],[a, t, r, y, w, s], X)
restituisce
yes X=[a, y, r]

Esercizio 4 (punti 5)

Dato il seguente albero di gioco, in cui il primo giocatore è MAX, si applichi l'algoritmo MIN MAX per decidere la prima mossa da effettuare e i tagli alfa beta. Si ricorda che i tagli si possono effettuare anche per valori correnti uguali ad alfa o beta rispettivamente.



Esercizio 5 (punti 6)

Si introduca l'algoritmo di unificazione. Si spieghi poi che cosa è l'occur check e che cosa implica la sua assenza nell'interprete del linguaggio Prolog.

SOLUZIONE

Esercizio 1:

Logica:

$\exists X$ (paz(X) and $\forall$ (Y med(Y) $\Rightarrow$ piace(X,Y)))
$\forall X \forall Y$ paz(X) and ciar(Y)$\Rightarrow$ not piace (X,Y)

Query: $\forall Y$ med(Y)$\Rightarrow$ not ciar(Y)
(va negata)

Clausole

1. paz(c)
2. piace(c,Y) or not med(Y)
3. not paz(X) or not ciar(Y) or not piace(X,Y)

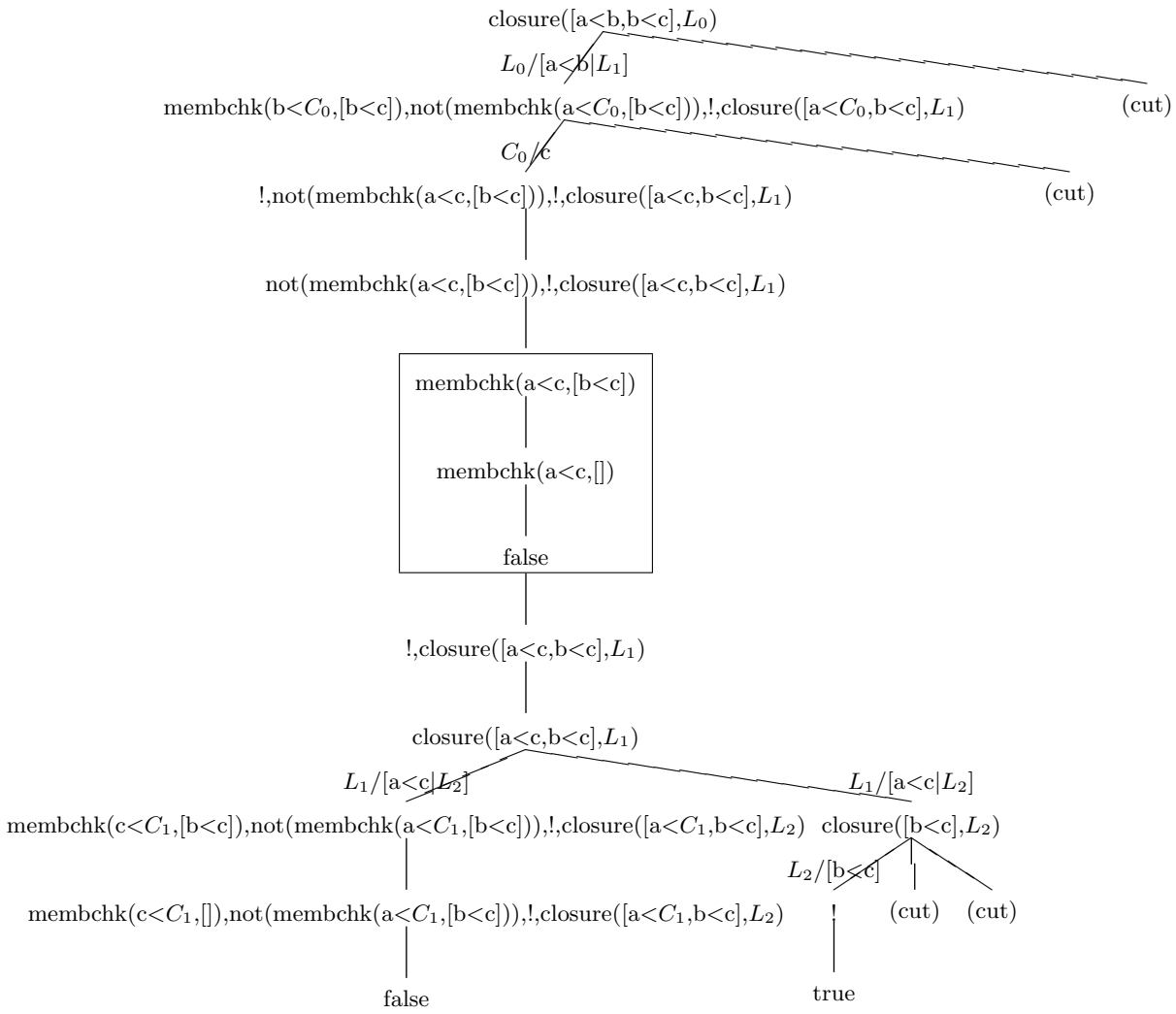
dalla Query negata:

4. med(d)
5. ciar(d)

Risoluzione:

6. da 1 e 3: not ciar(Y) or not piace(c,Y)
7. da 5 e 6 not piace(c,d)
8. da 2 e 7: not med(d)
9. da 4 e 8 : [].

Esercizio 2:

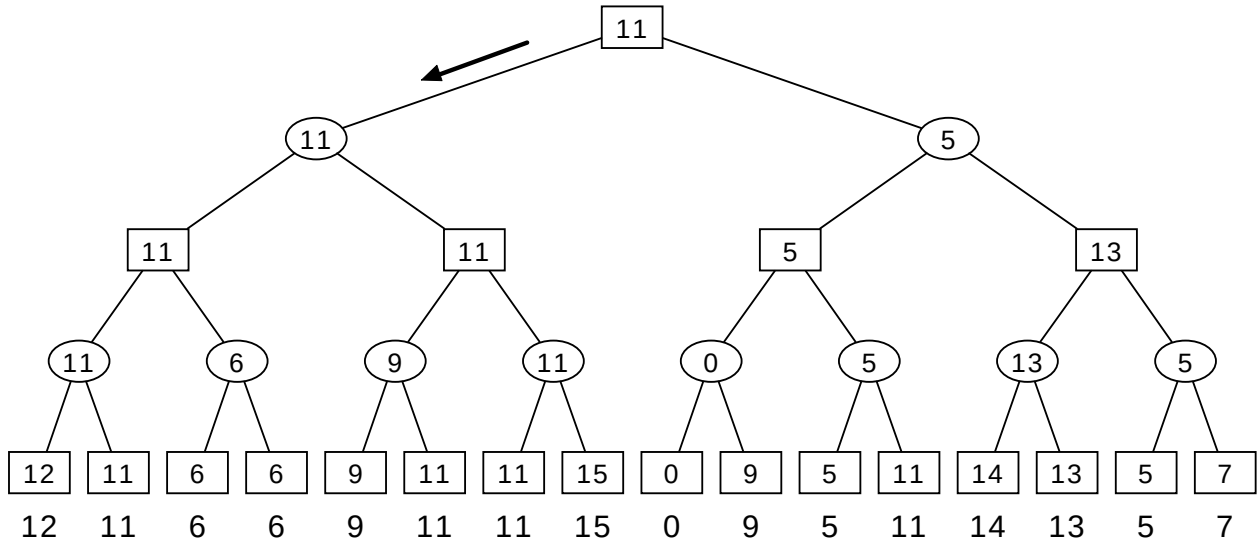


Esercizio 3:

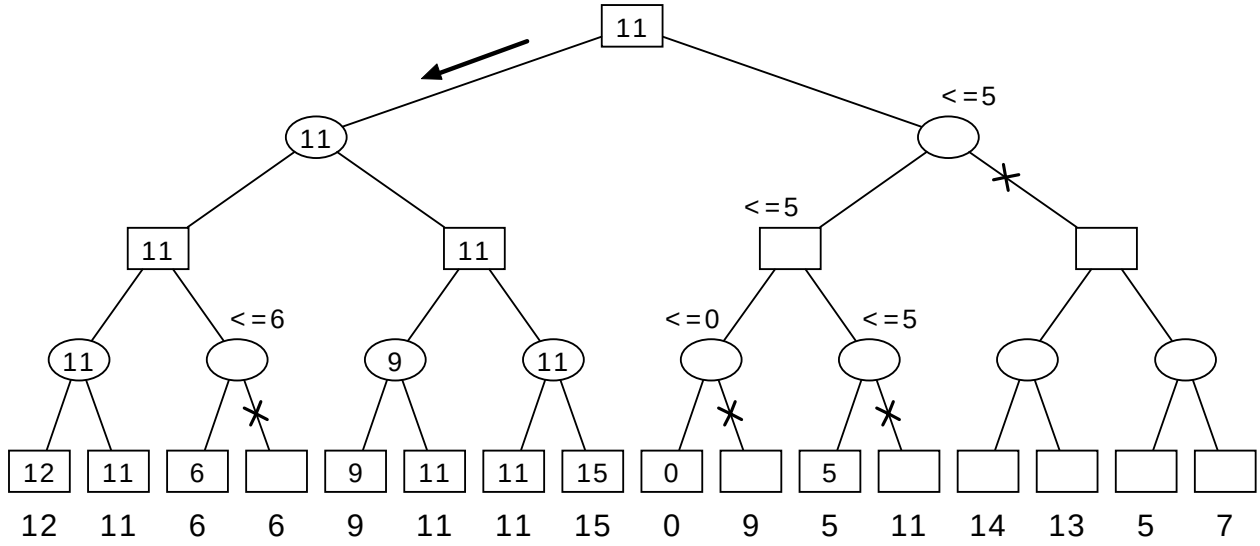
```
listpart([], L,  []):- !.
listpart([N1|Tail], L, [A|B]):- count(N1,L, A),
listpart(Tail, L, B).


count(1,[A|_],A):-!.
count(N, [_|B], C):- N >1, N1 is N-1, count(N1,B,C).
```

Esercizio 4:

Min max


11
11
5
11
11
5
13
11
6
9
11
0
5
13
5
12
11
6
6
9
11
11
15
0
9
5
11
14
13
5
7
12 11 6 6 9 11 11 15 0 9 5 11 14 13 5 7

Alfa beta


11
<=5
11
<=5
11
11
<=6
<=0
<=5
11
9
11
12
11
6
9
11
11
15
0
5
12 11 6 6 9 11 11 15 0 9 5 11 14 13 5 7