**FONDAMENTI DI INTELLIGENZA ARTIFICIALE – PRIMA PARTE (6 CFU)**

**9 Dicembre 2010 – Tempo a disposizione 2h – Risultato 32/32 punti**

**Esercizio 1 (punti 6)**

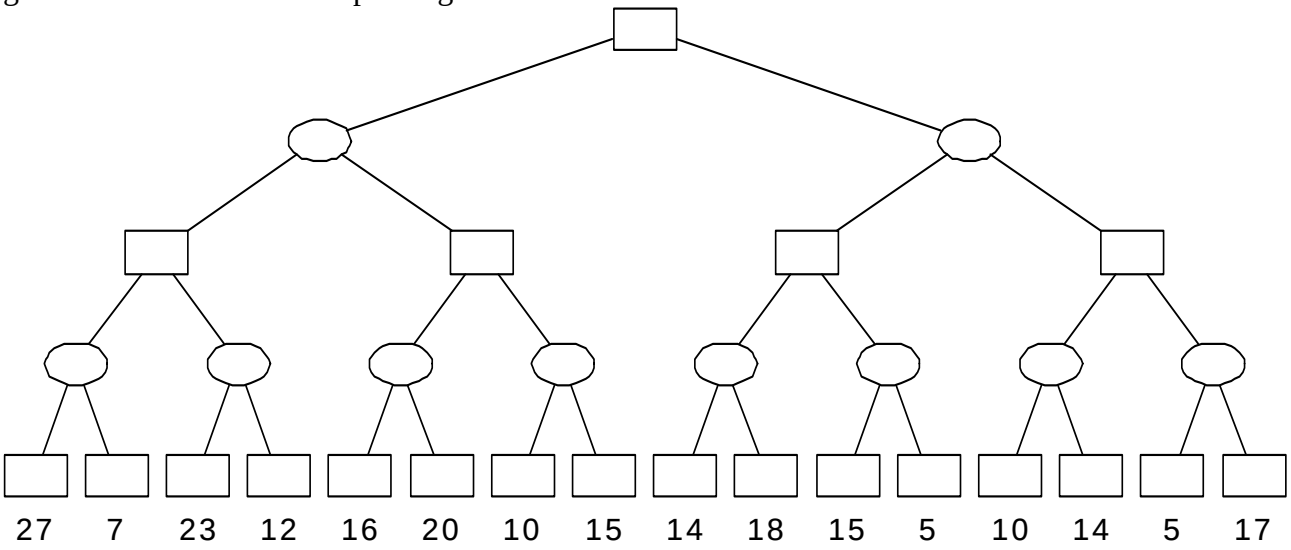
Si rappresentino in logica dei predicati del primo ordine le seguenti asserzioni:

- *Maria e Carlo sono due persone*
- *Maria ama lo sci*
- *Maria ama il teatro*
- *Carlo ama il teatro*
- *Due persone si amano se amano una stessa cosa*

- Applicando il principio di risoluzione dimostrare che esiste una persona che Carlo ama.
- Rappresentare in Prolog le frasi precedenti e scrivere poi la chiamata al meta-predicato Prolog che consente di collezionare in una lista tutte le cose che "maria" ama.

**Esercizio 2 (punti 4)**

Si consideri il seguente albero di gioco in cui la valutazione dei nodi terminali è dal punto di vista del primo giocatore. Si assuma che il primo giocatore sia Max.




Si mostri come gli algoritmi min-max e alfa-beta risolvono il problema

**Esercizio 3 (punti 6)**

Il seguente predicato Prolog:

```
listup(X,X,[X]).
listup(Low,Up,[Low|L]):-
    X is Low+1,
    listup(X,Up,L).
```

dovrebbe creare una lista che contiene tutti i numeri da un minimo a un massimo; ad esempio `listup(1,5,L)` fornisce `L/[1,2,3,4,5]`. Il predicato però ha un ramo infinito: si modifichi il programma, inserendo un cut, per evitare che ci siano rami infinit. Si disegni poi l'albero SLD relativo alla query `?-listup(1,3,[A,B,C,D|L])`.

**Esercizio 4 (punti 3)**

Si scriva un programma Prolog `somma/2` che data una lista L di numeri interi ed un numero N e ha successo se la somma degli elementi di L vale N. Esempio:

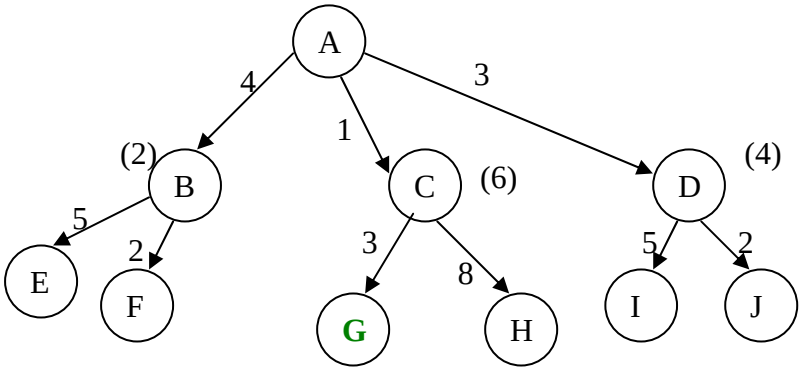
```
?- somma([1,2,3,4,5,6], 21).
Yes
?- somma([1,2,3,4], N).
Yes N=10
```

**Esercizio 5 (punti 5)**
Il Tricolore: sette bambini stanno discutendo sui tre colori di una bandiera che hanno visto e ciascuno di loro è "sicurissimo" di ricordarne due con precisione:

- Bianco e Giallo
- Giallo e Nero
- Azzurro e Rosso
- Verde e Azzurro
- Marrone e Giallo
- Marrone e Verde
- Nero e Rosso

In realtà, della coppia di colori ricordata da ciascun bambino, solo uno dei due colori era davvero presente.
Si modelli il problema come CSP, con variabili che possono assumere i valori [0,1].
Si risolva il problema utilizzando l'euristica *Minimum Remaining Value* per la scelta della variabile da istanziare (a parità di valori, si considerino le variabili da istanziare secondo il loro ordinamento alfabetico) e i valori da provare secondo l'ordine [0,1], applicando il Forward Checking dopo ogni labeling.

**Esercizio 6 (punti 6)**
Si consideri il seguente albero di ricerca in cui i numeri sugli archi sono i costi e quelli (tra parentesi) vicino agli stati le stime euristiche *h*. Ove *h* non è specificata, vale 0.




Si assuma che i nodi siano espansi in ordine alfabetico, nel caso in cui la strategia di ricerca applicata li consideri equivalenti. Il goal sia lo stato G e la ricerca termini al suo raggiungimento.
Si indichi la lista di stati espansi da ognuna delle seguenti strategie di ricerca:

- Breadth First
- Depth First
- Iterative Deepening Search *(facoltativo)*
- Best-First search
- A* search

Si indichi inoltre se la funzione *h* è ammissibile in questo esempio. E' consistente? Si motivi la risposta data.

**Esercizio 7 (punti 2)**
Dare la definizione di correttezza e completezza (per la FOL – First Order Logic). Discutere queste due proprietà per il linguaggio Prolog (indicare se valgono o meno, e perché).

VOTO:
Esame da 6 CF, il voto è determinato da questa prima parte
Esame da 9 CFU, è la media pesata della I parte (che vale 2/3) e della II ( che vale 1/3) ovvero il voto finale è dato da: ((voto_Iparte +voto_IIparte)/3)*2 e varia quindi da 0 a massimo 32 (equivalente a lode).

**FONDAMENTI DI INTELLIGENZA ARTIFICIALE – SECONDA PARTE (3 CFU)**

**9 Dicembre 2010 – Tempo a disposizione 30'+15' – Risultato 16/16 punti**

**Algoritmi genetici (punti 3)**

Descrivere il ciclo secondo cui operano gli algoritmi genetici, e i principali operatori utilizzati.

**Metainterpreti (punti 10)**

Si costruisca un meta-interprete per Prolog (in Prolog) che chieda all'utente i goal che non è in grado di dimostrare con le clausole del programma, ma solo se essi sono *ground* (si supponga dato un predicato `ground/1` che è vero se il suo argomento è *ground*).
Se l'utente risponde `true`, il goal deve essere poi asserito come fatto *ground*.

**Esercizio DCG (punti 3) (solo per chi non ha svolto l'esercitazione in itinere)**

Data la seguente grammatica:

***P* = *{***
***<E> ::= <T> + <E> | <T> - <E> | <T>***
***<T> ::= <F>***
***<F> ::= <intero>***
***<intero> ::= …***
***}***

Di che tipo di grammatica si tratta?
Scrivere il programma DCG per il parsing di espressioni aritmetiche generabili dalla grammatica.
Si utilizzi il predicato SICStus **integer(X)** che è vero se **X** è un intero

Esempio:

```
?-expr([2+34+25-27],[]).
Yes
?-expr([2++25-27],[]).
No
```

**SOLUZIONE – PARTE I**

**Esercizio 1**

persona(maria) and persona(carlo)
ama(maria,sci)
ama(maria,teatro)
ama(carlo, teatro)
$\forall X \forall Y ( \exists Z$ (persona(X) and ama(X,Z) and persona(Y) and ama(Y,Z) )$\rightarrow$ama(X,Y) )
Query: $\exists Z$ persona(Z) and ama(carlo,Z)

*Clausole:*

1. persona(maria)
2. persona(carlo)
3. ama(maria,sci)
4. ama(maria,teatro)
5. ama(carlo, teatro)
6. ama(X,Y) or not persona(X) or not ama(X,Z) or not persona(Y) or not ama(Y,Z)
7. not ama(carlo,Z) or not persona(Z)

*Risoluzione:*

Da 4+5+1+2 in più passi:

8. ama(carlo,maria)

Da 8+7+1 in più passi si deriva la clausola vuota.

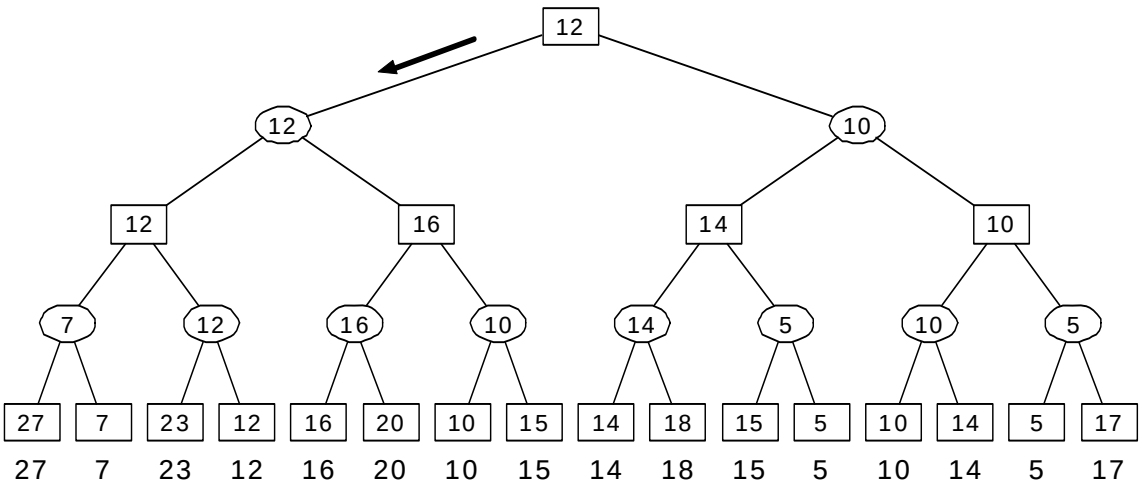
*Programma Prolog:*

```
ama(X,Y):- ama(X,Z),ama(Y,Z).
ama(maria,sci).
ama(maria,teatro).
ama(carlo, teatro).
persona(maria).
persona(carlo).
?-findall(Z,ama(maria,Z)L).
```

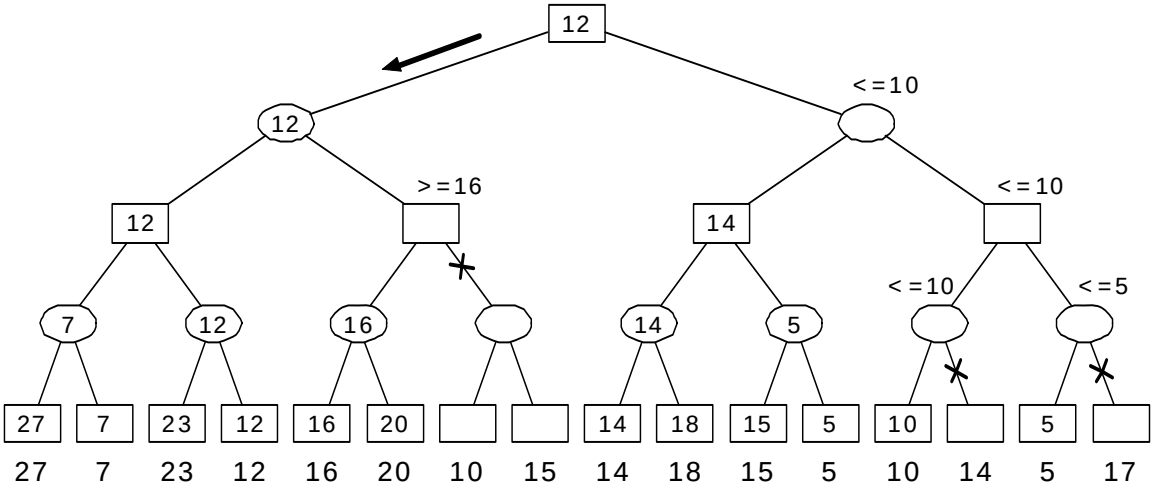
(chiamata; in realtà il programma Prolog presenta un loop)

**Esercizio 2**

min-max:



Alfa-beta:

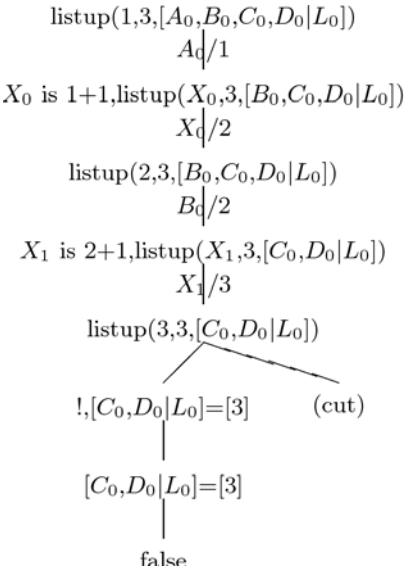



**Esercizio 3**

Predicato corretto:

```
listup(X,X,L):-!,L=[X].   %occorre spostare l'unificazione dalla testa al body
listup(Low,Up,[Low|L]):-
    X is Low+1,
    listup(X,Up,L).
```

Albero SLD:




Si noti che e` errato `listup(X,X,[X]):- !.`
in quanto l'unificazione del parametro di output L viene fatta prima del cut.

**Esercizio 4**

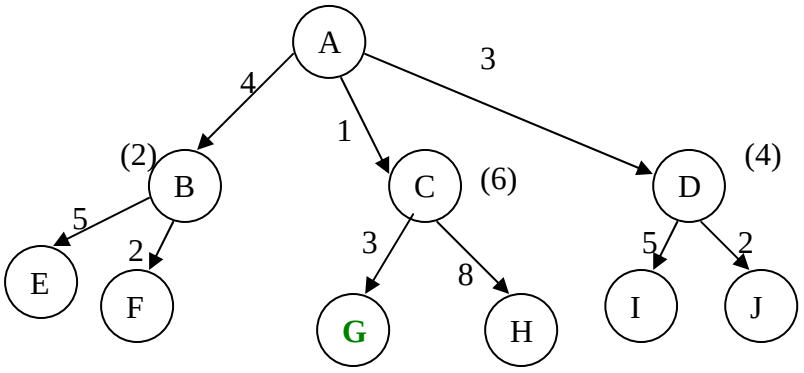
```
somma([],0):-!
somma([H|T],N):- somma(T,N1),
                 N is N1+H.
```

**Esercizio 5**
Variabili:
A, B, G, M, N, R, V ::[0,1]
Vincoli (solo una delle due è vera, se uso variabili intere posso rappresentare l'or esclusivo come somma con valore 1):
B+G=1 (oppure B=/=G)
G+N=1
A+R=1
V+A=1
M+G=1
M+V=1
N+R=1

| labeling | A=0 | B::[0,1] | G::[0,1] | M::[0,1] | N::[0,1] | R::[0,1] | V::[0,1] |
|---|---|---|---|---|---|---|---|
| FC | | | | | | R::[1] | V::[1] |
| labeling | | | | | | R=1 | |
| FC | | | | | N::[0] | | |
| labeling | | | | | N=0 | | |
| FC | | | G::[1] | | | | |
| labeling | | | G=1 | | | | |
| FC | | B::[0] | | M::[0] | | | |
| labeling | | B=0 | | | | | |
| labeling | | | | M=0 | | | |
| labeling | | | | M=0 | | | V=1 |

La soluzione è: Giallo, Rosso, Verde.

**Esercizio 6**




| | |
|---|---|
| Breadth First | A B C D E F **G** |
| Depth First | A B E F C **G** |
| Iterative Deepening Search | A A B C D A B E F C **G** |
| Best-First search | A B E F D I J C **G** |
| A* search | A B F C **G** |

La funzione *h* non è ammissibile in questo esempio perché non è sempre minore del valore costo effettivo (si veda il nodo C in cui l'euristica vale 6, ma la distanza dal goal è 3).
Non è nemmeno consistente, inoltre, perché per ogni nodo non vale $h(n) <= c(n,a,n') + h(n')$ (non è decrescente). Ad esempio, h(E) vale 9, mentre c(B,E)+h(B) vale 6).

**SOLUZIONE – PARTE II**

**Metainterpreti**

```
solve(true):-!.
solve((X,Y)):- !,solve(X),solve(Y).
solve(X):- clause(X,Body),solve(Body),!.
solve(X):-
     ground(X),
     ask(X,Risp),Risp==true, asserta(X).

ask(X,Risp):-
     write(X),
     write("  true or false ? "),
     read(Risp).


% oppure:
solve(true):-!.
solve((A,B)):-!,solve(A),solve(B).
solve(A):-clause(A,B),solve(B),!.
solve(A):- ground(A),write(A), write("  true or false ? "),
           read(R),R==true,asserta(A),!.
```

**SOLUZIONE – DCG**

```
expr --> term.
expr --> term, [+], expr.
expr --> term, [-], expr.
term --> factor.
factor --> [I], {integer(I)}.
```