Corso di Paradigmi
Prova scritta del 21 settembre 2010.

Tempo a disposizione: ore 2.

1. Con la notazione $\mathcal{C}^L_{L_1,L_2}$ indichiamo un compilatore da L_1 a L_2 scritto in L. Con $\mathcal{I}^L_{L_1}$ indichiamo un interprete scritto in L per il linguaggio L_1; se P è un programma in L_1 e x un suo dato, $\mathcal{I}^L_{L_1}(P,x)$ indica l'applicazione dell'interprete a P e x. Si dica se la seguente scrittura ha senso
   $\mathcal{I}^L_L(\mathcal{C}^L_{L,L_1}, \mathcal{C}^{L_1}_{L,L_1})$.
   Se la risposta è "no", si motivi tale fatto; se è "sí" si dica qual è il risultato ottenuto.

2. Si diano almeno tre esempi di vincoli sintattici contestuali (detti anche vincoli di semantica statica), cioè vincoli sintattici non esprimibili mediante grammatiche libere.

3. È data la grammatica $G = (\{S,T,a,b,+,*\}, \{a,b,+,*\}, S, P)$ dove l'insieme P delle produzioni è costituito da
   $$\begin{array}{lcl} S & \rightarrow & S+S \mid T \\ T & \rightarrow & T*T \mid (S) \mid a \mid b \end{array}$$
   Si dia un albero di derivazione per ciascuna delle stringhe seguenti: $a+b*a+b$ e $a+(b*a)+b$. Si dica qual è la precedenza indotta dalla grammatica tra gli operatori + e * .

4. Si presenti *sinteticamente* la tecnica del *reference count* per ovviare al problema delle *dangling references*; si discutano vantaggi e svantaggi.

5. Si dica cosa viene stampato dal seguente frammento di codice scritto in uno pseudo-linguaggio che usa scoping statico e passaggio di parametri per nome.

```
int x=2
void foo (nome int y){
   x = x + 1;
   y = y + 20;
   x = x + y;
   write (x);
}
{ int x=50
  foo (x);
  write (x);
}
```

6. Si consideri il seguente codice Java:

```
class A{
   int x=1;
   int f (int y){return g()+y;}
   int g(){return x;}
   }
class B extends A{
   int x=3;
   int g (){return x;}
   }
A a = new B();
int n = a.f(4) + a.x;
```

   Qual è il valore di `n` al termine del frammento?

7. Il linguaggio imperativo `Ric` è costituito dagli usuali comandi (assegnamenti, controllo di sequenza ecc.), non permette comandi di allocazione (e deallocazione) esplicita della memoria, ammette funzioni, ma, nel caso di funzioni ricorsive, queste devono essere ricorsive in coda. Si dica, motivando la risposta, qual è la più semplice forma di gestione della memoria utilizzabile nell'implementazione di `Ric`.

8. Si consideri il seguente frammento in un linguaggio con eccezioni e passaggio per valore-risultato:

```
{int y=0;
 void f(value-result int x){
    x = x+1;
    throw E;
    x = x+1;
    }
 try{ f(y); } catch E {};
 write(y);
}
```

Cosa viene stampato ?