Corso di Paradigmi di Programmazione
Prova scritta del 19 Settembre 2006.

Tempo a disposizione: ore 2.

1. È dato l'alfabeto (di terminali) $T = \{(,), \bullet, 0\}$; si considerino ora la grammatica $G_1 = (\{A\}, T, A, P_1)$ con P_1 dato da

$$A \quad ::= \quad A \bullet A \mid (A) \mid 0$$

e la grammatica $G_2 = (\{A, B\}, T, A, P_2)$, con P_2 dato da

$$\begin{array}{lll} A & ::= & A \bullet B \mid B \\ B & ::= & (A) \mid 0 \end{array}$$

Le due grammatiche generano lo stesso linguaggio? Motivare.
Una delle due grammatiche è ambigua. Quale? Motivare.

2. 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
$\mathcal{I}^L_L(\mathcal{C}^L_{L,L_1}, \mathcal{I}^L_{L_1})$.
Se la risposta è "no", si motivi tale fatto; se è "sí" si dica qual è il risultato ottenuto.

3. Si dica cosa stampa il seguente frammento in uno pseudolinguaggio con passaggio per riferimento e scope statico

```
int x = 2;
void foo(reference int y){
    x = x+1;
    y = y+10;
    x = x+y;
    write(x);
}
{int x = 10;
 foo(x);
 write(x);
}
```

4. Il linguaggio imperativo `Fun` è costituito dagli usuali comandi (assegnamenti, controllo di sequenza ecc.), non permette comandi di allocazione (e deallocazione) esplicita della memoria, ma ammette funzioni di ordine superiore (una funzione, cioè, può essere sia argomento che risultato di un'altra funzione). Si dica, motivando la risposta, qual è la più semplice forma di gestione della memoria utilizzabile nell'implementazione di `Fun`.

5. Si consideri il seguente frammento in uno pseudolinguaggio con tipi statici, dove `f` è una certa funzione di due argomenti:

```
int i,j;
float y,z;
y = f(i,j);
z = f(y,i);
```

Si fornisca (i) una possibile intestazione per la funzione `f` e (ii) le ipotesi che occorre fare sul sistema di tipi dello pseudolinguaggio affinché l'intestazione data in (i) sia corretta.

6. Si assuma di avere uno pseudolinguaggio che adotti la tecnica del *reference count*; se *OGG* è un generico oggetto nello heap, indichiamo con `OGG.ref-c` il suo reference count (nascosto). Si consideri il seguente frammento di codice:

```
class C { int n; C next;}
C foo = new C();  // oggetto OG1
C bar = new C();  // oggetto OG2
foo.next = bar;
bar.next = foo;
foo = bar;
```

Si dia il valore di OG1.ref-c e OG2.ref-c dopo l'esecuzione del frammento. Quali di questi due oggetti possono essere restituiti alla lista libera?

7. Si consideri il seguente frammento di codice in un linguaggio con scope statico nel quale il passaggio di parametri avviene per nome.

```
{int x = 10;
int w = 9;

int fie(int x,w){
     x = w + x
     w = w + x;
      }

{ x=3;


   ******

write(x); }
}
```

   Si scriva al posto degli asterischi una chiamata di fie tale che il valore scritto dal seguente comando write(x) sia 12, se questo è possibile. Altrimenti si motivi la risposta.

8. Si definiscano in Java due tipi `Sup` e `Sot`, tali che (i) `Sup` sia un supertipo di `Sot` e (ii) *non* vi sia ereditarietà tra `Sup` e `Sot`.