Corso di Paradigmi di Programmazione
Prova scritta del 1 febbraio 2012.

Tempo a disposizione: ore 2. **Svolgere gli esercizi 1-4 e 5-8 su fogli differenti.**

1. Si dice in genere che Java è un linguaggio interpretato. Si spieghi sinteticamente, ma con precisione, cosa si intende dire con tale espressione.

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. Si dica se la seguente scrittura ha senso

   $\mathcal{I}^{L}_{L_1}(\mathcal{I}^{L_1}_{L}, (\mathcal{C}^{L}_{L,L_1}(P^{L})))$.

   Se la risposta è "no" si motivi tale fatto; se è "sí", si dica qual è il risultato ottenuto.

3. Data una grammatica $G = (NT, T, S, P)$ si dia con precisione la definizione di "*linguaggio generato da G*".

4. Si consideri un linguaggio con scope statico per tutti i nomi, nel quale i nomi delle eccezioni devono essere dichiarati con la sintassi

   `exception` *`nomeeccezione`*`;`

   Qual è l'effetto dell'esecuzione del seguente frammento?

```
exception X;

void f(){
    throw X;
 }

void B(){
    exception X;
    void g(){
        try {f();} catch (X){
                     print("in␣g");
                     }
    }
}
B();
```

5. Si dica cosa viene stampato dal seguente frammento di codice scritto in uno pseudo-linguaggio che usi scoping dinamico e deep binding:

```
int x = 3;
procedure stampa_x(){
    write_integer(x);
    }
procedure ass_x(n:int){
    x = n;
    if (n=1) stampa_x();
    }
procedure pippo(function S, P ; int n){
    int x = 10;
    if (n=1) then  ass_x(n)
             else {S(n);
                   P();
             }
    }
pippo(ass_x, stampa_x, 1);
pippo(ass_x, stampa_x, 2);
```

6. Usando uno pseudolinguaggio con puntatori si fornisca un frammento di codice che generi un "dangling reference." Si faccia vedere come con la tecnica delle "tombstone" il problema è rimosso.

7. Si consideri l'implementazione dello scope statico mediante display. Si dica, motivando la risposta, se dimensione massima del display può essere determinata durante la compilazione.

8. Si supponga che "B_exp" e "Stat" siano non terminali di una grammatica libera già definiti e che **if, then, endif, else** siano terminali. Sono ora dati due linguaggi A e B, in tutto identici tranne che nella definizione del comando condizionale, che corrisponde al non terminale "IfThenE". Il linguaggio A contiene la definizione:

   IfThenE ::= **if** B_exp **then** Stat **endif** | **if** B_exp **then** Stat **else** Stat **endif**

   Il linguaggio B contiene la definizione

   IfThenE ::= **if** B_exp **then** Stat | **if** B_exp **then** Stat **else** Stat

   Si motivino le risposte alle seguenti domande: 1) Le due regole definiscono la stessa sintassi (ossia, $A = B$ come linguaggi formali)? 2) Supponendo che entrambi i linguaggi possiedano anche la regola

   Stat :: = IfThenE

   ci sono dei vantaggi nella scelta di A o di B?