Corso di Paradigmi di Programmazione
Prova scritta del 08.07.2003.

Tempo a disposizione: ore 2.

1. Si discuta *sinteticamente* la differenza tra sottotipi ed ereditarietà in un linguaggio ad oggetti. Si esemplifichi in Java.

2. Nella definizione del linguaggio di programmazione **LdC**, release 1.0, sono presenti le seguenti clausole in BNF (simboli terminali in neretto; Com è il simbolo iniziale):

   Exp → **a**
   Com → **skip** | ITE
   ITE → **if** Exp **then** Com | **if** Exp **then** Com **else** Com

   Si diano due alberi di derivazione diversi per una stessa stringa.

   Nella release 2.0, i progettisti di **LdC** modificano la terza clausola in

   ITE → **if** Exp **then** Com **fi** | **if** Exp **then** Com **else** Com **fi**

   Si giustifichi la modifica apportata.

3. Si fornisca un esempio di oggetto denotabile la cui vita sia più lunga di quella dei legami (nomi, puntatori, ecc.) che vi si riferiscono.

4. Si consideri il seguente frammento di programma scritto in uno pseudo-linguaggio che usi scoping dinamico e dove la primitiva `read(Y)` permette di leggere nella variabile `Y` un intero dall'input standard, mentre `write(X)` permette di stampare il valore della variabile `X`.

```
...
int X;
X = 1;
int Y;

void fie {
   foo;
   X = 0;
   }

void foo {
   int X;
   X = 5;
   }

read(Y);

if Y > 0 then { int X;
                X = 4;
                fie;
              }
         else { fie;
              }

write(X);
```

   Si dica quali sono (o qual è) i valori stampati.

5. Si descriva brevemente la struttura di un record di attivazione (o frame) di un linguaggio a blocchi con scoping statico.

6. Sono date le seguenti definizioni di tipo in un linguaggio di programmazione che usa equivalenza strutturale (dichiarazioni trasparenti):

```
type T1 = record {
           int a;
           bool b;
           };

type T2 = record {
           int a;
           bool b;
           }

type T3 = record {
           T2 u;
           T1 v;
           }

type T4 = record {
           T1 u;
           T2 v;
           }
```

Nello scope delle dichiarazioni `T3 a; T4 b;` è ammesso l'assegnamento `a = b;` ? Giustificare brevemente.

7. Il seguente frammento di codice viene fornito ad un *interprete* di uno pseudo-linguaggio che ammette parametri per *costante* (o read-only):

```
int X = 2;
void foo (constant int Y){
    write(Y);
    Y=Y+1;
    }
foo(X);
write(X);
```

Si dica qual è l'effetto dell'interpretazione.

8. È dato il seguente programma in uno pseudo-linguaggio che ammette eccezioni:

```
void foo() throws Exc {
     throw new Exc();
     }

void h (int X) throws Exc {
   if (X==1) {foo();}
   try {foo();} catch (Exc p) {**GESTORE1**}
   }

...
int Y;
read(Y);
try {h(Y);} catch (Exc p) {**GESTORE2**}
```

È possibile che venga eseguito `GESTORE2`? In caso positivo, in quali condizioni?