Corso di Paradigmi di Programmazione
Prova scritta del 8 Settembre 2004.

Tempo a disposizione: ore 2.

1. Come può essere partizionato l'ambiente di un linguaggio di programmazione moderno? Si descrivano brevemente i tipi di ambiente elencati.

2. Si fornisca una grammatica (libera) che generi il linguaggio $\{a^n b^m \mid n, m \geq 1\}$ usando solo produzioni della forma $N \rightarrow tM$ oppure $N \rightarrow t$ dove N e M sono qualsiasi simboli non terminali e t è un qualsiasi simbolo terminale. **Facoltativo:** Si dica se esiste una grammatica con tali caratteristiche che generi il linguaggio $\{a^n b^n \mid n \geq 1\}$ fornendo una spiegazione intuitiva della risposta.

3. 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 = new C();    // oggetto OG3
bar = foo;
```

   Si dia il valore di OG1.ref-c, OG2.ref-c e OG3.ref-c dopo l'esecuzione del frammento. Quali di questi tre oggetti possono essere ritornati alla lista libera?

4. Si dica cosa viene stampato dal seguente frammento di codice scritto in uno pseudo-linguaggio che usa scoping statico e passaggio per nome. (Si ricordi che un comando della forma `foo(w++);` passa a foo il valore corrente di `w` e poi incrementa `w` di uno).

```
{int x = 2;

  void pippo(name int y){
       x = x + y;
       }

   { int x = 5;
     { int x = 7
     }
     pippo(x++);
     write(x);
   }

write(x); }
```

5. Si fornisca un (breve) esempio di codice nel quale due successive chiamate della stessa procedura, nello stesso punto del programma e con gli stessi parametri attuali, producono risultati diversi.

6. In uno pseudolinguaggio con eccezioni (`try/catch`) si incontra il seguente blocco di codice:

```
public static void ecc() throws X {
   throw new X();
}
public static void g (int para) throws X {
   if (para == 0) {ecc();}
   try {ecc();} catch (X) {write(3);}
}
```

```
public static void main () {
   try {g(1);} catch (X) {write(1);}
   try {g(0);} catch (X) {write(0);}
}
```

Si dica cosa viene stampato all'esecuzione di `main()`.

7. Sono date le seguenti definizioni Java:

```
interface A {
    int val=1;
    int foo (int x);
}
interface B {
    int z=1;
    int fie (int y);
}
class C implements A, B {
    int val = 2;
    int z =2;
    int n = 0;
    public  int foo (int x){ return x+val+n;}
    public int fie (int y){ return z+val+n;}
}
class D extends C {
    int val=3;
    int z=3;
    int n=3;
    public int foo (int x){return x+val+n;}
    public int fie (int y){ return z+val+n;}
}
```

Si consideri ora il seguente frammento di programma

```
int u, v, w, z;
A a;
B b;
D d = new D();
a = d;
b = d;
System.out.println(u = a.foo(1));
System.out.println(v = b.fie(1));
System.out.println(w = d.foo(1));
System.out.println(z = d.fie(1));
```

Si dia il valore di `u`, `v`, `w` e `z` al termine dell'esecuzione.

8. **Solo per: corso AL; corso MZ a.a. 2002/03** Supponiamo di rappresentare i naturali usando 0 per lo zero e s(N) per il successore di N. Si dica qual'è la risposta calcolata dal seguente programma logico per un generico goal p(s,t,X) dove s e t sono termini che rappresentano nueri naturali (come detto sopra) e X è una variabile.

```
p(0,X, X).
p(s(Y),X, s(Z)):- p(Y,X,Z).
```

9. **Solo per il corso MZ a.a. 2003/04:** Si descrivano le seguenti strategie di valutazione nel caso di un linguaggio di programmazione funzionale: (i) in ordine applicativo (o call-by-value, o eager) e (ii) in ordine normale (o call-by-name). Si dia poi un esempio di programma (in Scheme, o λ-calcolo) per il quale le due strategie danno risultati diversi.