Corso di Paradigmi di Programmazione
Prova scritta del 21 Settembre 2004.

Tempo a disposizione: ore 2.

1. Si diano le definizioni di ambiente, scope e tempo di vita, descrivendo le loro relazioni reciproche.

2. Si fornisca una grammatica (libera) che generi il linguaggio $\{a^n bc^m \mid n, m \geq 1\}$ 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.

3. Si assuma di avere uno pseudolinguaggio che adotti la tecnica del *locks and keys*. Se p è un generico puntatore, indichiamo con `p.key` la sua key (nascosta); analogamente se OGG è un generico oggetto nello heap, indichiamo con `OGG.lock` il suo lock (nascosto). Si consideri il seguente frammento di codice, dove `free(p)` indica la deallocazione esplicita dell'oggetto riferito dal putatore `p`:

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

   Per tutti i puntatori coinvolti, si diano possibili valori per le keys; per ogni oggetto coinvolto, si diano possibili valori per i locks. Si dica poi, motivando la risposta, qual è un possibile risultato dell'esecuzione del codice: `foo.n = 1; foo.next.n = 0;`.

4. Si dica cosa viene stampato dal seguente frammento di codice scritto in uno pseudo-linguaggio che usa scoping dinamico e passaggio per riferimento.

   ```
   {int x = 1;
   int  y = 1;

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

      { int y = 3;
        { int x = 3
        }
        pippo(y);
        write(y);
      }
   write(y); }
   ```

5. Si consideri lo schema di codice seguente, nel quale vi sono due "buchi" indicati rispettivamente con (*) e (**). Si dia del codice da inserire al posto di (*) e (**) in modo tale che:

   (a) se il linguaggio usato adotta scoping statico, le due chiamate alla procedura `foo` assegnino a `x` lo stesso valore;

   (b) se il linguaggio usato adotta scoping dinamico, le due chiamate alla procedura `foo` assegnino a `x` valori diversi.

   La funzione `foo` deve essere opportunamente dichiarata in (*).

```
{int i;
(*)
for (i=0; i<=1; i++){
   int x;
   (**)
   x= foo();
   }
}
```

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

```
public static void ecc(int para1)  {
   try {throw new X();} catch (X) {write(1);}
  }
public static void f (int para2) {
  if (para2 == 1) {ecc(1);}
  try { ecc(2);} catch (X) {write{2);}
  }

public static void main () {
    try {f(1);} catch (X) {write(3);}
    try {f(2);} catch (X) {write(4);}
}
```

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

7. Si dica quali dichiarazioni Java comportano la definizione di un nuovo tipo.

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 e supponiamo di poter usare una primitiva `write(x)` che stampa il termine `t`. Si scriva un programma logico che stampi tutti i numeri naturali

9. **Solo per il corso MZ a.a. 2003/04:** Si discutano vantaggi e svantaggi del paradigma di programmazione funzionale.