Corso di Paradigmi di Programmazione
Prova scritta del 20 Gennaio 2004.

Tempo a disposizione: ore 2.

1. Si descriva brevemente cosa è una chiusura e a che cosa serve.

2. Si fornisca una grammatica che generi il linguaggio $\{a^n bc^n | n \geq 0\}$.

3. Si dica cosa viene stampato dal seguente frammento di codice scritto in uno pseudo-linguaggio che usa scoping statico e passaggio di parametri per valore. Si suppone che la primitiva `write(x)` permette di stampare un valore intero.

```
{int x;
 x = 2;

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


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

write(x); }
```

4. Si consideri il seguente frammento di codice in uno pseudo-linguaggio che ammetta passaggio dei parametri per riferimento e per nome.

```
int[] V = new int[5];
int n=0;

int f (reference int x) {
    return x++; }

void foo(reference int x, name int y){
    x++; y++; x++; y++;}

V[0]=V[1]=V[2]=V[3]=V[4]=1;

foo(V[f(n)], V[f(n)]);
```

   Si dia lo stato del vettore `V` al termine dell'esecuzione del codice esposto (si ricordi che un comando della forma `return w++;` restituisce il valore corrente di `w` e poi incrementa `w` di uno).

5. 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:

```
C foo = new C();  // oggetto OG1
C bar = new C();  // oggetto OG2
C fie = foo;
bar = fie;
```

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

6. Si scriva un frammento di codice, nel linguaggio di programmazione preferito, che generi un *dangling reference*.

7. **Solo per il corso AL:** Si consideri il seguente frammento di codice in un linguaggio nel quale il passaggio di parametri avviene per nome.

```
{int x,v;
 x=7;
 w = 2;

int fie(int w,x){
     y = 10;
     w = w + x;
     }


******

write(x); }
```

   Si scriva al posto degli asterischi una chiamata di fie tale che il valore scritto dal seguente comando write(x) sia 10.

8. **Solo per il corso MZ:** Si consideri la seguente definizione Java del metodo `foo` (in una qualche classe `A` non significativa per il problema):

```
int foo (C x, C y) {
   x.a = 0;
   y.a = 0;
   x.a = 1;
   if (x.a == y.a) return 1;
    else return 0;
   }
```

   Si diano:

   (a) una dichiarazione per la classe `C`;
   (b) la dichiarazione di una o più istanze di `C` ed una chiamata del metodo `foo` che ritorni il valore 1.

9. **Solo per il corso AL:** Si consideri il seguente programma logico

```
p(X):- q(a), r(Y).
q(b).
q(X):- p(X).
r(b).
```

   Si dica se il goal p(b) termina o meno, giutificando la risposta (ricordiamo che X,Y sono variabili e a,b costanti).

10. **Solo per il corso MZ:** Un nostro amico ci ha detto che la funzione Scheme `OMEGA`, quando chiamata sull'argomento 1, va in ciclo. Si consideri ora il seguente programma (`#t` è il valore booleano "vero"):

```
(DEFINE FIE
    (LAMBDA (X) (IF #t 1 X)))
(FIE (OMEGA 1))
```

    Qual è il suo comportamento in Scheme? Quale sarebbe il suo comportamento se Scheme adottasse una valutazione in ordine normale?