Corso di Paradigmi di Programmazione
Prova scritta del 10 gennaio 2008.

Tempo a disposizione: ore 2.

1. In un certo linguaggio di programmazione LP, un *identificatore* è una stringa su $\{a, b, c\}$, che inizia per a ed è una palindrome. Si dia una grammatica il cui linguaggio generato è costituito da tutti e soli gli identificatori di LP.

2. Un certo programma P prende come input un compilatore per il linguaggio $\mathcal{L}$ scritto in $\mathcal{H}$ e restituisce un compilatore per il linguaggio $\mathcal{L}$ scritto in $\mathcal{L}$. Si dica, con precisione, che tipo di programma è P. In quale linguaggio è scritto?

3. Il passaggio *per necessità* (*by need*) è una variante del passaggio per nome che vuole ovviare all'inefficienza della valutazione multipla del parametro attuale in corrispondenza di occorrenze multiple del parametro formale. Il passaggio per necessità è in tutto analogo al passaggio per nome, se non che, quando il parametro formale viene incontrato per la prima volta durante l'esecuzione, il corrispondente attuale viene valutato ed il suo valore *salvato*. Quando l'esecuzione richiede una nuova valutazione del parametro formale, viene usato il valore precedentemente salvato.

   Si mostri che il passaggio per nome e quello per necessità sono semanticamente diversi. Più precisamente, dato il seguente frammento:

   ```
   int x = 2;
   void foo(name/need int y){
       write(y+y);
   }
   foo(???);
   ```

   si dia un'espressione che, usata come parametro attuale al posto di `???`, faccia sì che `foo` stampi un valore quando il passaggio avviene per nome, e stampi invece un altro valore quando il passaggio avviene per necessità.

4. Si consideri il seguente frammento in uno pseudolinguaggio con scope statico e parametri di ordine superiore:

   ```
   {
   int x = 100;
   void foo (void f(), int n){
       void fie(){
         write(n+x);
       }
       if (n==0) f();
       else foo(fie,0);
    }
    {  int x = 5;
       void g(){
          write(x);
       }
       foo(g,1);
    }
   }
   ```

   Si dica cosa stampa il frammento con (i) shallow binding; (ii) deep binding.

5. Sono date le seguenti definizioni di funzione:

   ```
   int fact (int n){
       if (n == 0) return 1;
       else return n*fact(n-1);
   }
   int loop (int n){
       return loop(n);
   }
   ```

   Una certa implementazione del linguaggio si comporta nel modo seguente. Alla chiamata `fact(-1)`, risponde dopo qualche tempo con `Stack overflow during evaluation`, abortendo l'esecuzione. Alla chiamata `loop(1)`, non risponde, rimanendo in un loop infinito. Si dia una spiegazione motivata di questi due fatti, apparentemente contrastanti.

6. È dato il seguente frammento di codice in uno pseudolinguaggio con scope statico gestito mediante display

```
void P1(){
   void P2(){
      corpo di P2
      }
   void P3(){
      void P4(){
         corpo di P4
         }
      corpo di P3
      }
   corpo di P1
   }
```

   Si descriva graficamente il display dopo la successione di chiamate (tutte attive) P1, P3, P4, P2, P3. Si descriva inoltre brevemente come viene determinato il puntatore al record di attivazione di P2 (da inserire nel display), quando P2 è chiamata da P4.

7. È dato il seguente frammento in uno pseudolinguaggio con garbage collector basato su contatori dei riferimenti.

```
class C {
   int n; C next;
   }
C f(){
   C x = new C();
   C y = new C();
   x.next = y;
   y.next = x;
   return x;
   }
void g(){
   C x = new C();
   C y = new C();
   x.next = y;
   y.next = x;
   }
C foo = f();
g();
```

   Quanti oggetti sono ancora allocati nello heap al termine del frammento?

8. Si consideri il seguente codice Java:

```
class A{
   int x;
   int f (int y){return y+1;}
   }
class B extends A{
   int y;
   void g (int z){...}
   }
class C extends B{
    int x;
   int f (int y){return y+2;}
   }
 C pippo = new C();
 A pluto = new A();
 A quo = pippo;
```

   Si supponga che la gerarchia delle classi sia implementata mediante vtable. Si mostri graficamente come sono implementati classi e oggetti creati (con i relativi puntatori).