Corso di Paradigmi di Programmazione
Prova scritta del 21 settembre 2005.

Tempo a disposizione: ore 2.

1. Con la notazione $\mathcal{C}^L_{L_1,L_2}$ indichiamo un compilatore da L_1 a L_2 scritto in L. Con $\mathcal{I}^L_{L_1}$ indichiamo un interprete scritto in L per il linguaggio L_1; se P è un programma in L_1 e x un suo dato, $\mathcal{I}^L_{L_1}(P,x)$ indica l'applicazione dell'interprete a P e x. Si dica in meno di 10 parole cosa è $\mathcal{I}^L_{L_1}(\mathcal{C}^{L_1}_{L_1,L_2},\mathcal{C}^{L_1}_{L_1,L_2})$.

2. Si dica, motivando la risposta, quali delle seguenti regole (produzioni) possono comparire in una grammatica libera da contesto (i nonterminali sono costituiti da singole lettere maiuscole; i terminali da singole lettere minuscole).

$$A \to, \quad ab \to A, \quad bB \to aA, \quad A \to aA, \quad aB \to aA, \quad AB \to aA$$

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 nome. La primitiva `write(x,y,z)` permette di stampare i valori dei tre argomenti.

```
{int x = 2;
 int y = 5;
 int z = 10;
void pippo(name int v, name int w){
     int x = 1000;
     w = v;
     v = v+w+z;
     z = 1000;
     }
   { int x = 20;
     int y = 50;
     int z = 100;
     pippo(x, y);
     write(x,y,z);
   }
write(x,y,z) }
```

4. Si consideri il seguente frammento in un linguaggio con eccezioni e passaggio per valore-risultato e per riferimento:

```
{int y=0;
 void f(int x){
    x = x+1;
    throw E;
    x = x+1;
    }
 try{ f(y); } catch E {};
 write(y);
}
```

   Si dica cosa viene stampato dal programma qualora il passaggio dei parametri avvenga: (i) per valore-risultato; (ii) per riferimento.

5. Si consideri l'iterazione determinata, realizzata mediante un comando della forma

```
for I = inizio to fine by passo do C
```

   dove l'indice `I` è una variabile che non può essere modificata (né esplicitamente, né implicitamente) nel corso dell'esecuzione del corpo `C`, mentre `inizio`, `fine` e `passo` sono delle costanti.

   Si dica, motivando la risposta, se un tale comando può essere espresso mediante un comando `while` e, vice versa, se un comando `while` può essere espresso da un comando di iterazione determinata nel contesto un linguaggio imperativo che abbia anche i comandi di assegnamento, sequenza e condizionale.

6. È dato il seguente frammento di codice in uno pseudolinguaggio con `goto`, scope dinamico e blocchi annidati etichettati (indicati con `A :{...}`):

```
A: { int x = 5;
     int y = 4;
     goto C;
        B: {int x = 4;
            int z = 3;
            goto E;
            }
        C: {int x = 3;
            D: {int x = 2;
                }
            goto B;
            }
         E: {int x = 1; // (**)
             }

     }
```

Lo scope dinamico è gestito mediante lista delle associazioni (A-list) Si illustri graficamente la situazione della A-list nel momento in cui l'esecuzione raggiunge il punto segnato con il commento `(**)`. Quale è un inconveniente di questo tipo di gestione ?

7. Si considerino le seguenti definizioni di tipi record

```
type S = struct{
   char d[10];
   int t;
   };
type T = struct{
   char nome[5];
   int n;
   S s;
   }
```

Si supponga che un `int` sia memorizzato su 4 byte e un `char` su un byte. In un architettura a 32 bit con allineamento alla parola, quanti byte sono usati per memorizzare una variabile di tipo `T`? Perché?

8. **Solo per il corso AL** È dato il seguente programma Prolog (ricordiamo che `X` e `Y` sono variabili mentre `a` e `b` sono costanti):

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

Si dica se il goal `p(X)` termina o meno, giustificando la risposta. Nel caso in cui termini che cosa viene calcolato ?

9. **Solo per il corso MZ** Si consideri la seguente definizione di classe generica in Java 5:

```
class Coppia<A,B>{
   private A a;
   private B b;
   Coppia(A x, B y){   //costruttore
      a=x; b=y;
      }
   ...
}
```

È data ora le seguente definizione di metodo generico

```
Coppia<Object,Object> diagonale(Object x){
      return new Coppia<Object,Object>(x,x);
      }
```

Si dica se le seguenti dichiarazioni sono accettate dal compilatore:

```
String v = new String("pippo");
Coppia<Object,Object> co = diagonale(v);
Coppia<String,String> cs = diagonale(v);
```