Corso di Paradigmi di Programmazione
Prova scritta del 14 gennaio 2010.

Tempo a disposizione: ore 2.

1. È data una grammatica con terminali $\{a, b\}$ e non-terminali $\{A, B\}$. Quali delle seguenti *non* possono essere produzioni di tale grammatica?

$$A \to aAa \qquad A \to A \qquad A \to AbB \qquad A \to AA \qquad AA \to A \qquad A \to B \qquad aA \to aA$$

2. 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. Infine se P^L è un programma scritto in L e x un suo dato di input, $\mathcal{I}^{L_1}_L(P^L, x)$ indica l'applicazione dell'interprete a P^L e x. Si dica se la seguente scrittura ha senso

   $\mathcal{I}^L_{L_1}(\mathcal{I}^{L_1}_L, (\mathcal{C}^L_{L,L_1}, P^L))$.

   Se la risposta è "no" si motivi tale fatto; se è "sí", si dica qual è il risultato ottenuto e se tale risultato sia ottenibile in modo più semplice (sotto le stesse ipotesi che permettono di ottenerlo con l'espressione precedente).

3. Cosa stampa il seguente frammento in un linguaggio con scope statico e passaggio dei parametri per riferimento?

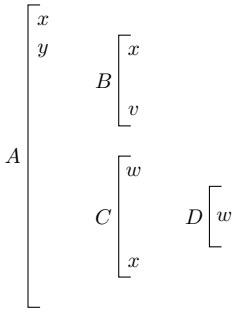
```
int x = 4;
void f (reference int z){
   int x = 0;
   void g (reference int w){
      x = z + w + 1;
      }
   z++;
   g(z);
   x++;
   write(x);
}
f(x);
write(x);
```

4. Si assuma di avere uno pseudolinguaggio che adotti la tecnica dei *locks and keys*. Se OGG è un generico oggetto nello heap, indichiamo con `OGG.lock` il suo lock (nascosto); se PTR è un generico puntatore (sulla pila o nello heap), indichiamo con `PTR.key` la sua key (nascosta). 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 diano possibili valori di `OG1.lock, OG2.lock, foo.key, fie.key` e `bar.key` dopo l'esecuzione del frammento.

5. Si consideri la struttura di blocchi schematizzata nella figura seguente; i nomi all'interno di un blocco indicano una dichiarazione di quel nome.



   L'ambiente segue la regola di scope dinamico. Si rappresenti graficamente la tabella centrale dell'ambiente (CRT) dopo la sequenza di chiamate A,B,C,D (tutte attive).

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

```
{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.

7. Si considerino le seguenti definizioni di classe in 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 f (int y){return y+2;}
   }
```

   Si supponga che la gerarchia delle classi sia implementata mediante vtable. Qual è la struttura della vtable di `C`?

8. Si consideri il seguente codice Java:

```
class A{
   int x=1;
   int f (int y){return g()+y;}
   int g(){return x;}
   }
class B extends A{
   int x=3;
   int g (){return x;}
   }
A a = new B();
int n = a.f(4) + a.x;
```

   Qual è il valore di `n` al termine del frammento?