Corso di Paradigmi di Programmazione
Prova scritta del 15 giugno 2011.

Tempo a disposizione: ore 2.

SCRIVERE LE SOLUZIONI A 1-4 E 5-8 SU DUE FOGLI DIVERSI

1. Si dimostri che la seguente grammatica è ambigua:

$$S \to Ab \mid A$$
$$A \to aAb \mid a \mid \varepsilon$$

2. Si diano almeno tre esempi di vincoli sintattici contestuali (detti anche vincoli di semantica statica), cioè vincoli sintattici non esprimibili mediante grammatiche libere.

3. Si dica, motivando la risposta, quali delle seguenti regole (o produzioni) possono comparire in una grammatica libera da contesto (i simboli maiuscoli sono non-terminali, quelli minuscoli sono terminali):

$$A \to B, \quad a \to A, \quad AB \to aA, \quad A \to AaA, \quad aB \to aA$$

4. Si dica cosa stampa il seguente frammento in uno pseudolinguaggio con passaggio per riferimento e scope statico

```
int x = 2;
void foo(reference int y){
    x = x+1;
    y = y+10;
    x = x+y;
    write(x);
}
{int x = 10;
 foo(x);
 write(x);
}
```

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

```
{void foo (int f(), int n){
    int m = 10;
    int fie(){
      write(n,m);
    }
    if (n==0) f();
    else {m = 30;
             foo(fie,0);
             }
 }
 int g(){
    write(10);
 }
 foo(g,1);
}
```

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

6. 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.

7. L'esecuzione del seguente frammento di codice su una certa implementazione risulta nella stampa del valore 14.

```
int V[10];
int x = 4;
for (int i=0, i<10, i++) V[i]=i;
V[x] = V[x++] + V[x++];
write (V[x]+ (x++));
```

   Si fornisca una possibile spiegazione.

8. Si considerino le seguenti definizioni di classe in un generico linguaggio orientato agli oggetti

```
class A{
   int x;
   int f (int y){return y;}
   }
class B extends A{
   int x;
   int y;
   void g (int z){...}
   }
class C extends B{
   int f (int y){return -y;}
   }
   C o = new C()
```

   Si descrivano brevemente (in modo grafico) due diverse implementazioni dell'oggetto `o` e dell'ereditarietà.