Corso di Paradigmi di Programmazione
Prova scritta del 18 Settembre 2007.

Tempo a disposizione: ore 2.

1. In un certo linguaggio di programmazione LP, un *identificatore* è una stringa su $\{a, b, 0, 1\}$, che inizia per a o b e può anche contenere, una sola volta, il carattere #. Si dia una grammatica il cui linguaggio generato è costituito da tutti e soli gli identificatori di LP.

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

```
int x = 2;
int y = 3;
void foo(name int y, name int z){
    x = x+1;
    y = y+10;
    z = x+y;
    write(x,z,y);
}
{int x = 10;
 foo(x,y);
 write(x);
}
```

3. In un certo linguaggio gli assegnamenti sono valutati nel seguente modo: prima l'espressione a destra del simbolo =; poi quella a sinistra; infine si esegue l'assegnamento. Si consideri il seguente frammento di codice:

```
int x = 1;
int A[5];
for (int i=0; i<5; i++)
   A[i] = i;
A[x++] = A[x++]+1;
```

   Qual è lo stato del vettore `A` dopo l'assegnamento ?

4. Cosa è una *dangling reference*? È possibile crearne una in Java? Motivare brevemente.

5. Si consideri la seguente dichiarazione di array multidimensionale

```
int x;
read(x);
int A[10][x];
```

   dove il comando `read(x)` permette di leggere il valore della variabile `x` dall'esterno.

   Come viene memorizzato l'array A ? Inoltre sappiamo che: un intero è memorizzato su 4 byte; l'array è memorizzato in ordine di riga, con indirizzi di memoria crescenti (cioè se un elemento è all'indirizzo i, il successivo è a $i + 4$ ecc.); il valore di x letto e' 5. Qual è l'offset dell'elemento `A[2][4]` rispetto all'inizio dell'array? (Si risponda in notazione decimale).

6. Si consideri il seguente frammento di codice

```
int x = 5;

void pippo(int x, int y){
    int z = 0;
    z = x+1-y+y;
    write(z);
}
{pippo (*, *);
write (x);
}
```

   Si definiscano le modalità di passagio di parametri di pippo e si indichino i parametri attuali da sostituire agli *, in modo tale che i valori stampati dal codice siano 7 e 8 (in quest'ordine).

7. Si considerino le seguenti dichiarazioni in Java:

```
class A{
   int x = 5;
   int f(int n){return x+2*n;}
}
class B extends A{
   int x = 2;
   int f(int n){return x+n;}
   }
A a = new B();
```

Nello scope di tali dichiarazioni, qual è il valore dell'espressione `a.f(1)` ?

8. In un certo linguaggio, il tipo T è compatibile col tipo S. Nello scope delle dichiarazioni:

```
S s;
T t;
int g(S x){...}
```

si consideri l'espressione

```
g(t) + g(s)
```

La funzione `g` è necessariamente *overloaded*? La macchina astratta inserisce in tale espressione coercizioni? In caso positivo, dove?