Corso di Paradigmi di Programmazione
Prova scritta del 14 febbraio 2006.

Tempo a disposizione: ore 2.

1. (i) Si dica, se la grammatica

$$G_1 = (\{S, T\}, \{a, b\}, S, \{S := aSb \mid ab \mid T,\ T := bTa \mid ba\})$$

può generare la stringa `abababababab`.

(ii) Se la risposta a (i) è si, si fornisca una derivazione per tale stringa, se è no, si dica come andrebbe modificata la grammatica per ottenere la stringa.

2. La tabella che segue riporta la semantica operazionale dei comandi di un semplice linguaggio imperativo (coincide con la Figura 2.14 del testo).

Si descriva la computazione corrispondente al comando

$$\textbf{while } \neg X == Y \textbf{ do } X := X + 1;$$

nello stato $\sigma = [(X, 4), (Y, 5)]$. Le espressioni aritmetiche e booleane possono essere valutate con il loro significato intuitivo.

3. Si considerino due linguaggi imperativi L1 ed L2 che non ammettono ricorsione. I comandi di L1 permettono assegnamento, comando condizionale ed iterazione determinata, quelli di L2 invece assegnamento, comando condizionale ed iterazione indeterminata. Si dica, motivando la risposta, se i due linguaggi hanno lo stesso potere espressivo oppure no.

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

```
int x = 3;
int y = 4;
void foo(reference int y, reference int z) {
    int x = 5;
    y = y+1;
    if (z==y) write(x);
    else write (y);
    }
foo(x,x);
write(x);
write(y);
```

(ii) Si dica poi cosa stampa lo stesso frammento se il passaggio dei parametri avviene per valore-risultato.

5. Si fornisca un frammento di codice nel quale, a seconda della valutazione lazy o eager delle espressioni si ottengono risultati diversi.

6. Si descriva la struttura delle vtable corrispondenti alle seguenti dichiarazioni di classi (si assuma ereditarietà singola):

```
class A{
   int a = 1;
   int f(){return 1;}
   int g(){return 2;}
}
class B extending A{
   int a = 2;
   int g(){return 3;}
   int h(){return 4;}
}
```

7. Si assuma un linguaggio che permette la dichiarazione di array con dimensione non nota staticamente, ma fissata al momento dell'elaborazione della dichiarazione; tutti gli array hanno tipo indice costituito da un intervallo di interi, con inizio a 0. È data la seguente dichiarazione:

```
int A[n][m][n+m];
```

Si dia la struttura del dope vector per `A`.

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

```
A: { int x = 1;
     int y = 2;
     goto C;
        B: {int x = 3;
            int z = 4;
            int y = 5;
            E: {int x = 6; // (**)
            }
        C: {int x = 7;
            D: {int x = 8;
                }
            goto B;
            }
     }
```

Lo scope statico è gestito mediante display. Si illustri graficamente la situazione del display (e della pila dei record di attivazione) nel momento in cui l'esecuzione raggiunge il punto segnato con il commento `(**)`.