Prova Scritta di Linguaggi di Programmazione I

06/04/2006

1. Con riferimento al seguente programma in Pascal, si rappresenti il P-code relativo al corpo della procedura `bubble`, cioè il P-code corrispondente alla sezione di codice all'interno del riquadro.

```
program esercizio;
  ...   ...
type sequenza = array[ 1 .. 100 ] of integer;
  ...   ...
procedure swap( var x, y: integer );  ...
begin  ...   ...  end;
procedure bubble( k, n: integer; var seq: sequenza );
```

```
begin
  if  k < n  then
    begin
      bubble( k+1, n, seq );
      if  seq[k] > seq[k+1]  then
        swap( seq[k], seq[k+1] )
    end
end;
```

```
begin  ...   ...  end.
```

2. Sia $C_{L_1}^{L_2 \to L_3}$ un compilatore da L_2 a L_3 scritto in L_1 e $I_{L_1}^{L_2}$ un interprete di L_2 scritto in L_1. Sia $[\![P]\!]$ la funzione calcolata dal programma P. Si dica cosa produce la valutazione $[\![I_{L_0}^{L_1}]\!]([\![I_{L_0}^{L_2}]\!](C_{L_2}^{L_2 \to L_1}, C_{L_2}^{L_3 \to L_0}), I_{L_3}^{L_1})$ e se tale risultato può effettivamente esistere.

3. Si mostri che la grammatica

$$C ::= \texttt{w} \mid \texttt{i}\ B\ \texttt{t}\ C\ \texttt{e}\ C \mid \texttt{i}\ B\ \texttt{t}\ C$$
$$B ::= \texttt{b}$$

è ambigua. Si dia una grammatica non-ambigua che genera lo stesso linguaggio.

4. Si mostri l'output del seguente frammento di programma in un linguaggio C-like con assegnamento che calcola *r*-value *dopo* *l*-value e indici dei vettori che iniziano da 0:

```
{
  char x[10] = il_proprio_cognome;
  int i = 2;
  char magic(int j, ref char y)
  {
      char c = x[j++] = x[j++];
      x[--j] = x[--j];
      write(y, j);
      return c;
  }
  write(magic(0, x[i++]));
  write(x[i], i);
}
```

5. Assumendo di utilizzare nell'Esercizio 8 la tecnica di implementazione con shallow binding, si mostri con un diagramma la situazione sullo stack nascosto e nel vettore centralizzato per le variabili del sistema, al momento immediatamente successivo alla chiamata di Q ed al passaggio dei parametri per Q (cioè immediatamente prima della esecuzione del comando `return`).

6. Una matrice quadrata M di ordine n si dice *convergente* con raggio r se il modulo della somma degli elementi di ogni riga, escluso quello sulla diagonale, è inferiore a r. Data la seguente definizione del tipo di dato astratto (polimorfo) *Matrici* **type** Mat a = [[a]] si scriva un predicato Haskell

conv :: Num a => Mat a -> a -> Bool che determina se una matrice (quadrata) è convergente con raggio r. Si implementino le matrici per righe come liste di liste.

Si scriva il programma con variabili anonime ove possibile.

7. Data la seguente definizione del tipo di dato astratto*Annotated Binary Search Tree*

```
data (Ord a) => ABST a = Void
     | Node Bal a (ABST a) (ABST a)
data Bal = Left | Bal | Right
```

un albero è detto AVL se la differenza fra le profondità dei sottoalberi destro e sinistro di un qualunque nodo è al massimo 1. Si scriva un predicato Haskell `isAVL` che dato un albero determina se è AVL e se le etichette Bal dei nodi sono consistenti con lo (s)bilanciamento (dove Left e Right si riferiscono al sottoalbero più profondo).

Si dia espressamente il tipo di `isAVL`.

8. Si mostri l'output del seguente frammento di programma espresso in un linguaggio C-like con scoping *dinamico*, *shallow binding* e valutazione delle espressioni da sinistra a destra:

```
{ int x = 5;
  int y = 7;
  {   int x = 3;
      int Q(name int w)
      {   return (w + y++);
      }
      P(x, y, Q);
      write(y++,x++);
  }
  proc P(ref int y, valueresult int z, int R(name int))
  {   z = y-- + R(++x + ++y);
      write(x,y,z);
      z = R(z++ + --y);
  }
  write(y,x);
}
```

$5, 5, 14, 22, 5, 23, 5$