Prova Scritta di Linguaggi di Programmazione I

18/12/2006

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

```
program esercizio;
  ...   ...
const N = ... ;
type  vector = array[ 0 .. N ] of real;
  ...   ...
procedure up( n: integer; var v: vector );

  procedure swap( var x, y: real );
    var t: real;
  begin  t := x;  x := y;  y := t  end;
```

```
  var  i: integer;
begin
  if  n > 1  then
    begin
      if  v[n div 2] < v[n]  then
        begin
          swap( v[n div 2], v[n] );
          up( n div 2, v )
        end
    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 $[\![I_{L_1}^{L_2}]\!](C_{L_2}^{L_3 \to L_2}, C_{L_3}^{L_4 \to L_3})$.

3. Sia G la grammatica individuata dalle produzioni

$$S ::= C \mid C\ S\ C \qquad\qquad C ::= \mathtt{a} \mid \mathtt{b}$$

pensate per generare il linguaggio $L := \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w = w^R\}$, dove w^R è la stringa w rovesciata. Si dia **sia** un esempio dell'albero di parsing di una stringa generata per sbaglio ($\in \mathcal{L}(G) \setminus L$) **che** l'esempio di una stringa non generata ($\in L \setminus \mathcal{L}(G)$). Qualora non fosse possibile dare un esempio si giustifichi il perchè.

4. Si mostri l'output del seguente frammento di programma in un linguaggio C-like con assegnamento che calcola *r*-value *dopo* *l*-value, valutazione argomenti chiamate da sinistra a destra e indici vettori inizianti da 0:

```
char x[10] = {il_proprio_cognome};
int  i=2, j=3, y[3] = {0,2,1};
#define ind(e) ( (y[e--])++ )

char mess(int i, ref char z)
{ char c = x[ind(i)] = x[ind(i)];
  write(y[0]+y[1]+y[2], z, i+j++);
  return c;
}
write(mess(i--, x[i--]));
write(x[i--], i, j);
```

5. Assumendo di utilizzare nell'Esercizio 8 la tecnica di implementazione con catena statica e "display", si mostri schematicamente la situazione sul display e sullo stack di sistema al momento in cui viene eseguita P.

Prova Scritta di Linguaggi di Programmazione I

18/12/2006

6. Molte tecniche sviluppate per la compressione di immagini si basano su una codifica ad albero chiamata "Quad Tree". Si codificano in questo modo immagini quadrate il cui lato sia una potenza di 2. Se l'immagine è omogenea (stesso colore) si codifica, indipendentemente dalle sue dimensioni, con una foglia contenente il colore. Se l'immagine è eterogenea allora si utilizza un nodo i cui figli contengono le codifiche dei quadranti superiore-sinistro, superiore-destro, inferiore-sinistro, inferiore-destro, rispettivamente. Usando il tipo di dato

   ```
   data (Eq a)=> QT a = C a | Q (QT a) (QT a) (QT a) (QT a)
   ```

   si scriva una funzione Haskell `quadMap` che data una funzione f e un QuadTree q determina il QuadTree che codifica l'immagine risultante dall'applicazione di f a tutti i pixel dell'immagine codificata da q.

   Attenzione a non generare nodi del tipo $\texttt{Q}(\texttt{C}\,x)(\texttt{C}\,x)(\texttt{C}\,x)(\texttt{C}\,x)$ al posto del solo $\texttt{C}\,x$.

   Si usino variabili anonime quando possibile.

   Si dia esplicitamente il tipo della funzione.

7. Si scriva una funzione Haskell `occ` che dato un QuadTree ed un colore determina il numero (minimo) di pixel di quel colore. Ad esempio

   ```
   let z = C 0; u = C 1; q = Q z u u u in occ (Q q (C 0) (C 2) q) 0
   ```

   restituisce 6 (visto che il QuadTree codifica almeno 16 pixel).

   Si usino variabili anonime quando possibile.

   Si dia esplicitamente il tipo della funzione.

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

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

$8, 22, -8, 22, 28, 5, 7$