Prova Scritta di Linguaggi di Programmazione I

12/01/2007

1. Con riferimento al seguente programma in Pascal, si rappresenti il P-code relativo al corpo della procedura `swap` e alla relativa invocazione, cioè il P-code corrispondente alla sezione di codice all'interno dei riquadri.

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

    procedure swap( i, j: integer );

      var t: real;
    begin  t := v[i];  v[i] := v[j];  v[j] := t  end;


  begin
    if  n > 1  then
      begin
        if  v[n div 2] < v[n]  then
          begin

            swap( n div 2, n );

            up( n div 2, v )
          end
      end
  end;

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

2. Sia $PE_{L_1}^{L_2}$ un valutatore parziale di L_2 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 $[\![PE_{L_0}^{L_1}]\!](PE_{L_1}^{L_2}, I_{L_2}^{L_3})$ e che relazione ha con le Proiezioni di Futamura.

3. Sia G la grammatica individuata dalle produzioni

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

pensate per generare il linguaggio $L := \{ww \in \{\mathtt{a},\mathtt{b}\}^* \mid w = w^R\}$, 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'evoluzione delle variabili e l'output del seguente frammento di programma in un linguaggio C-like con assegnamento che calcola *l*-value *dopo* *r*-value, valutazione argomenti chiamate da sinistra a destra e indici vettori inizianti da 0:

```
char x[10] = {il_proprio_cognome};
int  i=0, 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. Utilizzando nell'Esercizio 8 la tecnica di implementazione con catena statica e display, si mostri schematicamente la situazione sul display e sullo stack di sistema quando entra in esecuzione F per la prima volta.

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) la 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 `flipHorAll` che data una lista di QuadTrees costruisca la lista dei QuadTrees che codificano le immagini ribaltate rispetto all'asse orizzontale.

Si usino variabili anonime quando possibile.
Si dia esplicitamente il tipo della funzione.

7. Si scriva una funzione Haskell `zipWith` per QuadTrees che, analogamente alla `zipWith` per le liste, data un'operazione binaria $\otimes$ e due QuadTrees q_1 e q_2 costruisce il QuadTree che codifica l'immagine risultante dall'applicazione di $\otimes$ a tutti i pixel della stessa posizione nelle immagini codificate da q_1 e q_2. Ad esempio

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

restituisce

```
Q (C 2) (C 3) (C 3) (C 3)
```

Si usino variabili anonime quando possibile.
Si dia esplicitamente il tipo della funzione.

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

```
int x = 5, y = 7;
int F(name int v, valueresult int z) {
  z += v; write(x,y,z);
  z += x--; write(x,z);
  return z--;
}
{   int y = 3, x = 1;
    int Q(ref int v, name int x) {
        int w = x;
        v -= F(y + v--, w);
        write(++v);
        return (w + ++y);
    }
    write(Q(y, F(x++, y) + y++);
}
write(x,y);
```

$5, 7, 5, 4, 10, 4, 7, 30, 3, 34, -24, 10, 3, 7$