Prova Scritta di Linguaggi di Programmazione I

17/09/2007

Si noti che quanto messo nei riquadri è una *bozza* fornita solo a *titolo indicativo*.
Quindi **non è** un modello di soluzione completa che ci si aspetta ad un esame.

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

```
program esercizio;
  const N = ...;
  var v: array [ 0 .. N ] of integer;
    ...    ...
  function choice( x, y: integer; b: boolean ) : integer;
  begin   if  b  then  choice := x  else  choice := y   end;

  function where( s, i, j: integer ) : integer;
    var k: integer;
  begin
    k := ( i + j ) div 2;
    if  v[k] = s  then
      where := k
    else
      where := where( s, choice(i,k+1,s<v[k]), choice(k-1,j,s<v[k]) )
  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 $[\![PE_{L_0}^{L_1}]\!](I_{L_1}^{L_2}, PE_{L_2}^{L_3})$.

3. Sia G la grammatica individuata dalle produzioni

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

ed $L := \{ww \mid vw = wv, v \in \{\mathtt{a},\mathtt{b}\}^+, w \in \{\mathtt{a},\mathtt{b}\}^*\}$. Si diano le stringhe di L di lunghezza ≤ 3 **e** un esempio dell'albero di parsing di una stringa generata per sbaglio ($\in \mathcal{L}(G) \setminus L$). Qualora non fosse possibile dare un esempio si giustifichi il perché.

$\{w \in L \mid |w| \leq 3\} = \{\varepsilon, \mathtt{aa}, \mathtt{bb}\}$.
$\mathtt{ab} \in \mathcal{L}(G) \setminus L$.

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 *prima* di *r*-value, valutazione argomenti chiamate *da destra a sinistra* e indici vettori inizianti da 0:

```
char x[10] = {il_proprio_cognome};
int  i=1, k=2, y[3] = {2,2,2};

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

Prova Scritta di Linguaggi di Programmazione I

17/09/2007

> Attenzione che l'ordine di valutazione degli argomenti delle chiamate *non ha nulla a che vedere* con quello che le procedure poi fanno con gli argomenti. In particolare una `write`(e_1, e_2) stamperà sempre prima (il valore di) e_1 e poi e_2.

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 seconda volta.

6. Rappresentando QuadTrees con il tipo di dato

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

si scriva un predicato Haskell `isRotatedIn` che dati un QuadTree t e una lista di QuadTrees ts determina se uno dei QuadTrees che codificano l'immagine di t ruotata di 0, 90, 180 o 270 gradi è un elemento della lista ts.

Si usino variabili anonime quando possibile.
Si dia esplicitamente il tipo di tutte le funzioni.

```
isRotatedIn x xs = any p xs
  where
    p y = y==x || y==x1 || y==x2 || y==x3
    x1 = rotate90Right x
    x2 = rotate90Right x1
    x3 = rotate90Right x2
```

con `rotate90Right` soluzione dell'Esercizio apposito dell'eserciziario.

7. Si scriva una funzione Haskell `difference` che dato un colore c ed un QuadTree q determina la differenza fra il numero di pixel dell'immagine codificata da q che hanno un colore maggiore di c e quelli minori di c. Ad esempio

```
let d = C 2; u = C 1; q = Q d u u u
in difference 1 (Q q (C 0) (C 3) q)
```

restituisce -4 (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'evoluzione delle variabili e l'output del seguente frammento di programma espresso in un linguaggio C-like con scoping *statico*, *deep binding*, assegnamento che calcola r-value *dopo* l-value e valutazione delle espressioni da sinistra a destra:

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

```
  y += z; write(v);
  return z--;
}
write(x,y);
```

$5, 6, 4, 1, 5, 9, 10, 5, -9, 12, 5, 19$