Prova Scritta di Linguaggi di Programmazione I

20/12/2007

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 funzione `search`, cioè la sezione di codice all'interno del riquadro.

```
program esercizio;
  ...   ...
var v: array [ 0 .. 255 ] of real;
  ...   ...
function middle( i, j: integer ) : integer;
begin  middle := (i + j) div 2  end;

function choose( left: boolean; i, j: integer ) : integer;
begin  if  left  then  choose := i  else  choose := j  end;

function search( x: real; i, j: integer ) : integer;
```

```
  var k: integer;
begin
  k := middle(i,j);
  if  x = v[k]  then
    search := k
  else
    search := search( x, choose(x<v[k],i,k+1), choose(x<v[k],k-1,j) )
end;
```

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

2. Sia $PE_{L_1}^{L_2}$ un valutatore parziale di L_2 scritto in L_1. Sia $[\![P]\!]$ la funzione calcolata dal programma P, $\forall P$. Si stabiliscano un programma Q scritto in L_2 ed un R per cui la valutazione $[\![PE_{L_4}^{L_3}]\!](R, Q)$ abbia senso e produca un interprete di L_1.

3. Sia L_1 il linguaggio $\{xyw \mid xw = w^R y, x, y \in \{\mathtt{b}, \mathtt{c}\}, w \in \{\mathtt{a}, \mathtt{b}\}^*\}$ e $L_2 := \{w \mid w = w^R, w \in \{\mathtt{b}, \mathtt{c}\}^*\}$, dove w^R è la stringa w rovesciata. Si diano le stringhe di L_1 ed L_2 di lunghezza ≤ 4.

   Inoltre si diano due grammatiche non ambigue, con simboli iniziali S_1 ed S_2 per generare L_1 ed L_2.

   Infine si dica se la grammatica ottenuta unendo le due precedenti e la produzione $S \rightarrow S_1 \mid S_2$ è ambigua (mostrando un testimone dell'ambiguità oppure argomentando opportunamente sulla non ambiguità).

$\{w \in L_1 \mid |w| \leq 4\} = \{\mathtt{bb}, \mathtt{cc}, \mathtt{bbb}, \mathtt{bbab}, \mathtt{bbbb}\}$.

$\{w \in L_2 \mid |w| \leq 4\} = \{\varepsilon, \mathtt{b}, \mathtt{c}, \mathtt{bb}, \mathtt{cc}, \mathtt{bbb}, \mathtt{bcb}, \mathtt{cbc}, \mathtt{ccc}, \mathtt{bbbb}, \mathtt{bccb}, \mathtt{cbbc}, \mathtt{cccc}\}$.

S ambigua: ad esempio si possono far vedere i due alberi di parsing di `bb`.

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:

```
int x[3] = {data_di_nascita};
int  i=2, k=0;
int mess(ref int z, int i) {
  x[i--] -= z++;
```

```
  write(i, (x[i++] += i *= x[k--]++));
  write(x[2]+x[0]);
  return x[k*i]-1;
}
write(mess(x[k++], i--));
write(x[i], i--);
```

> 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. Assumendo di utilizzare nell'Esercizio 8 la tecnica di implementazione CRT con pila nascosta, si mostri schematicamente la situazione sullo stack nascosto e nel vettore centralizzato quando entra in esecuzione F.

6. Si scriva un predicato `isSymmetric` che, data una matrice quadrata implementata come liste di liste per righe, determina se è simmetrica.

7. Rappresentando BST (Binary Search Tree) con il tipo di dato

```
data (Ord a,Show a) => BST a =
                Void | Node a (BST a) (BST a)
```

   si scriva una funzione Haskell `maxDiameter` che data una lista l di BST determina il massimo dei diametri dei BST di l. Il diametro di un BST è la lunghezza del massimo cammino fra due nodi, indipendentemente dall'orientamento degli archi.

   A titolo di esempio,

```
let
 t = Node 'f' (Node 'c' t1 t2) Void
 t1 = Node 'b' (Node 'a' Void Void) Void
 t2 = Node 'd' Void (Node 'e' Void Void)
in maxDiameter [t]
```

   vale 4

8. Si mostri l'evoluzione delle variabili e l'output del seguente frammento di programma espresso in un linguaggio C-like con scoping *dinamico*, assegnamento che calcola l-value *dopo* r-value e valutazione delle espressioni da sinistra a destra:

```
int x = 7, y = 5;
int Q(name int v, ref int x) {
    int w = F(v-x, y);
    y += w+x--; write(y);
    return (++x + w);
}
int F(name int v, valres int z) {
  y += x++; write(x,y,z);
  z -= y; write(v);
  return z--;
}
{   int x = 1, y = -2, z = 3;
    write(Q(x++ + z++, y));
    write(x+2,y);
}
write(x,y);
```

$-3, -3, -2, 7, 0, 2, 4, 1, 7, 5$