Prova Scritta di Linguaggi di Programmazione I

07/09/2009

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. Si rappresenti il P-code relativo alla funzione `manh`, cioè la sezione di codice Pascal all'interno del riquadro.

```
program esercizio;
  const IMAX = 9; UNKNOWN = 0;
  type harray = array[ 0..IMAX, 0..IMAX ] of integer; history = ^harray;
  ...   ...

  procedure init( i, j: integer; var ha: harray );
    var u, v: integer;
  begin
    for u := 0 to i do
      for v := 0 to j do ha[u,v] := UNKNOWN;
  end;

  function mmem( i, j: integer; h: history ) : integer;
  begin
    if h^[i,j] = UNKNOWN then
      begin
        if i*j = 0 then h^[i,j] := 1
                   else h^[i,j] := mmem( i-1, j, h ) + mmem( i, j-1, h )
      end;
    mmem := h^[i,j];
  end;

  function manh( i, j: integer ) : integer;
```

```
    var h: history;
  begin
    new( h ); init( i, j, h^ );
    manh := mmem( i, j, h )
  end;
```

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

```
manh:   ENT     1               IND                     LDA     0     5
        LDA     0     6         LDA     0     6         IND
        LDC   int   100         IND                     LDA     0     6
        NEW                     CUP     3  init         IND
        MST     1               LDA     0     0         CUP     3  mmem
        LDA     0     4         MST     1               STO
        IND                     LDA     0     4         RETF
        LDA     0     5         IND
```

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 $[\![[\![C_{L_4}^{L_5 \to L_6}]\!](I^{L_7})]\!](P, X)$ specificando i linguaggi in cui devono essere scritti I^{L_7} e P affinchè tale valutazione abbia senso.

Prova Scritta di Linguaggi di Programmazione I

07/09/2009

3. Sia G la grammatica individuata dalle produzioni

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

pensata per generare il linguaggio $L := \{v \mid v \in \{\mathtt{a},\mathtt{b}\}^*,\ v\mathtt{a} = \mathtt{a}v^R,\ \#(\mathtt{a}, v) > \#(\mathtt{b}, v)\}$, dove w^R è la stringa w rovesciata e $\#(x, w)$ è il numero di occorrenze del simbolo x dentro w.

La grammatica G è ambigua?

Si diano le stringhe di L di lunghezza ≤ 4; l'albero di parsing di una stringa di $\mathcal{L}(G) \setminus L$; una stringa di $L \setminus \mathcal{L}(G)$. Qualora non fosse possibile si giustifichi il perchè.

> $\{w \in L \mid |w| \leq 4\} = \{\mathtt{a}, \mathtt{aa}, \mathtt{aaa}, \mathtt{aaaa}, \mathtt{aaba}\}$.
> $\varepsilon \in \mathcal{L}(G) \setminus L$. $\mathtt{aa} \in L \setminus \mathcal{L}(G)$.

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, argomenti chiamate *da destra a sinistra* e indici vettori inizianti da 0:

```
write(mess(v[j],j--) + j);
write(v[i], i++);
int mess(ref int z, ref int i) {
 do  write( (v[j] += v[i++]--) )
 while ( ++v[i] < 70 );
 v[j--] -= z++;
 return i+j;
}
int v[3]={data_di_nascita},  i=0, j=1;
```

> 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 $\mathtt{write}(e_1, e_2)$ stamperà sempre prima (il valore di) e_1 e poi e_2.

5. Assumendo di utilizzare nell'Esercizio 8 la tecnica di implementazione del "display" si mostri (schematicamente) con dei diagrammi la situazione sul display e sullo stack di sistema quando entra in esecuzione H. La situazione incontrata è consistente con quanto dovrebbe succedere?

> La situazione *non* è consistente, H dovrebbe vedere F, non G.

6. Sia data la seguente definizione del tipo di dato astratto (polimorfo) *Weighted Binary Search Tree* che consiste in un BST in cui in ogni nodo viene mantenuta l'altezza del nodo stesso.

```
data (Ord a) => WBST a = Void | Node a Int (WBST a) (WBST a)
```

Si scriva una funzione `insert` che inserisce un nuovo valore in un WBST.

```
insert x Void = Node x 0 Void Void
insert x (Node y d l r)
  | x <= y     = Node y (max d (1+nld)) nl r
  | otherwise = Node y (max d (1+nrd)) l nr
    where
      nl = insert x l
```

```
        (Node _ nld _ _) = nl
        nr = insert x r
        (Node _ nrd _ _) = nr
```

7. Rappresentando Alberi "generici" con il tipo di dato

   ```
   data (Eq a) => Tree a = V | N a [Tree a]
   ```

   si scriva un predicato `iscorrect` che determina se un albero è un albero di parsing secondo le regole di una grammatica codificata mediante una funzione che, dato un simbolo, restituisce la lista delle possibili espansioni (stringhe di simboli) secondo le produzioni. Ad esempio è True

   ```
   let r S = [ [], [A, S] ]
       r A = [ [] ]
   in iscorrect r (N S [N A [], N S []])
   ```

   Assumiamo ovviamente di avere Tree ben formati in input

   ```
   iscorrect rules t =
     treefoldr aggr1 Nothing aggr2 (Just []) t /= Nothing
     where
       aggr2 (Just x) (Just xs) = Just (x:xs)
       aggr2 _          _       = Nothing

       aggr1 x (Just xs) | any (xs==) (rules x) = Just x
       aggr1 _ _                                = Nothing
   ```

   con `treefoldr` soluzione dell'apposito esercizio dell'eserciziario.

8. Si mostri l'evoluzione delle variabili e l'output del seguente frammento di programma espresso in un linguaggio C-like con scoping *specificato a programma*, *deep binding*, assegnamento che calcola *r*-value *dopo* *l*-value, valutazione delle espressioni da destra a sinistra e indici vettori inizianti da 0:

   ```
   int x = 2, y= 5, v[] = {9,8,7};
   static-scope int F(valres int w) {
     static-scope int H(ref int w) {
       w -= y-- + x++;
       write(x,y);
       return y++;
       }
     int x = w++;
     G( w-- + x++, H );
     return w++;
   }
   dynamic-scope void G(name int w, int R(ref int)) {
     int y = w;
     write(R(x) + x++,v);
     write(++x,y);
   }
   write(F(v[x++]));
   write(x,y,v);
   ```

$-5, 4, 12, 9, 8, 7, -4, 15, 7, 3, 5, 9, 8, 8$