27 Giugno 2014

Linguaggi di Programmazione

Cognome e nome	
Matricola	

1. Dato un alfabeto $\Sigma = \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$, si chiede di specificare l'espressione regolare delle stringhe che contengono al più due **b** non consecutivi.

2. Specificare la grammatica BNF di un linguaggio in cui ogni frase è un programma specificato da un linguaggio funzionale per la manipolazione di interi, come nel seguente esempio:

```
a=3, b=5, c=12;

function alfa(n, m) = a + b - 3,
function beta(x, y, z) = alfa(x + y - (a + 3), 5) - z + 1,
function gamma(w) = beta(w, a - 1, c * w) - (a + b) * w;

(a + b) * alfa(a, 3) - gamma(beta(a,b,c)).
```

Il programma è composto da tre sezioni, di cui solo la terza è obbligatoria. La prima sezione specifica un insieme di costanti. La seconda sezione specifica una serie di funzioni definite da una lista (anche vuota) di parametri formali ed una espressione (corpo della funzione). La terza sezione specifica l'espressione del programma. Una espressione coinvolge le quattro operazioni aritmetiche (con possibilità di parentesi) e chiamate di funzione.

3. È data la seguente tabella degli operatori (con precedenza decrescente verso il basso):

Operatori	*Associatività*	*Ordine di valutazione*
*** , /**	sinistra	Da sinistra a destra
+, -	sinistra	Da sinistra a destra
=	non associativo	Da sinistra a destra
and	destra	Da destra a sinistra
or	destra	Da destra a sinistra

e la seguente espressione: `a + b - c * d / e = (a + b) * c + d and d = e or a = b`

Assumendo che solo l'operatore **and** sia valutato in corto circuito, si chiede di:

1. Inserire le parentesi nella espressione, indicando così l'associazione degli operandi agli operatori;
2. Rappresentare l'albero della espressione;
3. Specificare la semantica operazionale della valutazione della espressione sulla base dei seguenti vincoli:

- Il linguaggio di specifica include tutti gli operatori indicati nella tabella, l'assegnamento (**:=**), la negazione logica (**not**), le costanti booleane, la selezione a più vie ed il **return**,
- Ogni operatore (ad eccezione dell'assegnamento) non può essere applicato al risultato di altre operazioni.

4. È dato il seguente frammento di grammatica BNF, relativo ad una lista di istruzioni in un linguaggio imperativo:

stat-list → *stat-list stat stat-list* | *stat*
stat → *def-stat* | *assign-stat*

Specificare la semantica denotazionale del corrispondente frammento di linguaggio, assumendo la disponibilità delle seguenti funzioni ausiliarie, che mappano una istruzione ed un certo stato in un nuovo stato (eventualmente **errore**): M_{def}(*def-stat*, `s`), M_{assign}(*assign-stat*, `s`). In particolare, specificare la funzione semantica M_{list}(*stat-list*, `s`).

5. Specificare nel linguaggio *Scheme* la funzione `swapairs`, avente in ingresso una lista (anche vuota) di un numero pari di elementi, la quale computa la lista con gli elementi scambiati a coppie, come nei seguenti esempi:

```
(swapairs '()) = ()
(swapairs '(2 5)) = (5 2)
(swapairs '(a b c d e f) = (b a d c f e)
(swapairs '(1 (2 3) (4 5 6) 7 () 8) = ((2 3) 1 7 (4 5 6) 8 ())
```

6. Specificare nel linguaggio *Haskell* la funzione `separa`, avente in ingresso una lista (anche vuota) di triple, la quale computa la corrispondente tripla di liste, come nei seguenti esempi:

```
separa [(1, 'a', True)] = ([1], ['a'], [True])

separa [(1, 'a', True), (2, 'b', False),(3, 'c', True)] =
       ([1, 2, 3], ['a', 'b', 'c'], [True, False, True])
```

7. Specificare in *Prolog* il predicato `swapairs(L,S)`, in cui `L` è una lista (anche vuota) di un numero pari di elementi, mentre `S` è la lista di elementi scambiati a coppie, come specificato al punto **5**.

8. Assumendo passaggio dei parametri per nome, spiegare l'effetto dell'esecuzione del seguente frammento di codice:

```
int i=0, a[4]={0,0,0,0};

void f(int n)
{
  n = 2;
  i++;
  n = 3;
  i--;
  n = 4;
}

main()
{
  f(a[i]);
}
```