21 Gennaio 2013

Linguaggi di Programmazione

Cognome e nome	
Email	

1. Specificare la definizione regolare relativa ai seguenti simboli lessicali di un linguaggio di programmazione:

- *Identificatore*: stringa alfanumerica, di lunghezza compresa tra due e quattro caratteri, che inizia con una lettera e termina con una cifra;
- *Costante intera*: sequenza di cifre priva di zeri non significativi;
- *Costante stringa* (anche vuota): racchiusa tra doppi apici e contenente qualsiasi carattere diverso da doppi apici e newline;
- *Costante booleana*: **true** o **false**;
- *Commento*: stringa di caratteri che inizia con # e termina con newline.

2. Specificare la grammatica EBNF di un linguaggio per la dichiarazione e assegnamento di variabili, come nel seguente esempio:

```
i, j: scalar;
alfa: record (x: scalar, y: array [3,7,10] of scalar);
vector: array[100] of scalar;
complex: array[2,5] of array[3] of record(a: scalar, b: scalar);
i = j + 1;
j = (i + j) * (i - j / 25);
alfa.x = alfa.y[2,4,6];
complex[1,3] = complex [1,4] - 15;
alfa.y[1,2,j+2] = vector[1 + j - vector[20] / alfa.x];
```

Ogni frase si compone di almeno una istruzione. Una variabile può essere scalare (cioè, intera) o strutturata. Esistono due costruttori di tipo (ortogonali fra loro): il record e l'array multidimensionale (in cui ogni dimensione viene specificata da un intero positivo). Le espressioni di assegnamento coinvolgono le quattro operazioni aritmetiche, le espressioni di indicizzazione degli array e l'estrazione di campi dei record, senza limiti di ortogonalità.

3. Specificare la semantica operazionale della espressione relazionale corrispondente alla seconda istruzione nel seguente frammento di programma (in cui **in** denota l'operatore di appartenenza):

```
R: table(a: integer, b: integer, S: table(c: integer));
select [ a = b and b in S ] R;
```

sulla base dei seguenti requisiti:

- L'operatore di congiunzione (**and**) ha precedenza minima ed è valutato in corto circuito;
- L'ordine di valutazione degli operandi è da sinistra a destra;
- Il linguaggio di specifica non dispone di un operatore di appartenenza;
- Il linguaggio di specifica dispone però della struttura di controllo **foreach x in X do** ... (per iterare su tutti gli elementi **x** di un insieme **X**).

4. È dato il seguente frammento di grammatica BNF, relativo alla specifica di un insieme di numeri:

set → { *numbers* }
numbers → **num** , *numbers* | **num**

Si chiede di specificare la semantica denotazionale del corrispondente frammento di linguaggio, definita come la somma di tutti i numeri positivi nell'insieme. Si assume la disponibilità della funzione ausiliaria M_n(**num**), che restituisce il valore lessicale di **num**.

5. Specificare nel linguaggio *Scheme* la funzione `medie`, avente in ingresso una lista di numeri, la quale restituisce una lista di lunghezza pari a quella in ingresso, in cui ogni elemento alla posizione *i*-esima è costituito dalla media dei numeri della lista di ingresso fino alla posizione *i*-esima, come nel seguente esempio:

```
(medie '(1 2 3 4 5)) = (1 1.5 2 2.5 3)
```

Nel caso di lista vuota, `medie` restituisce la lista vuota.

6. Specificare nel linguaggio *Haskell* la classe di tipi `Expr`, nella quale è definita la funzione binaria `somma`. Quindi, istanziare la classe `Expr` mediante i seguenti tipi:

- `Int`: in cui somma è la classica somma aritmetica tra interi;
- `[Int]`: in cui `somma` genera una lista di interi di lunghezza minima fra i due operandi, in cui ogni intero nella posizione *ì*-esima del risultato corrisponde alla somma aritmetica dei due interi nella posizione *ì*-esima nei rispettivi operandi, come nel seguente esempio:

  ```
  somma [1,2,3] [4,5,6,7] = [5,7,9]
  ```

- `[[Int]]`: in cui `somma` genera una lista di liste di interi di lunghezza minima fra le due liste operando, in cui ogni lista nella posizione *ì*-esima del risultato corrisponde alla concatenazione delle due liste nella posizione *ì*-esima nei rispettivi operandi, come nel seguente esempio:

  ```
  somma [[1,2],[3,4,5],[6,7]] [[8,9],[10]] = [[1,2,8,9],[3,4,5,10]].
  ```

7. Specificare nel linguaggio *Prolog* il predicato `uguali(X,Y)`, in cui `X` ed `Y` denotano una lista (istanziata) di interi che rappresenta un multi-insieme (cioè, un insieme con possibili duplicati, come ad esempio `[1,3,2,4,3,5,2,3,6,4]`), il quale risulta vero quando `X` ed `Y` includono gli stessi numeri ed ogni numero compare lo stesso numero di volte (eventualmente in posizioni diverse) nelle due liste.

8. Dopo aver illustrato il significato del predicato *cut* nel linguaggio *Prolog*, si analizzai il seguente programma.

```
alfa(10).
beta(20).
beta(30).
gamma(20).
gamma(30).
gamma(40).
omega(N) :- gamma(N), !, beta(N).
omega(N) :- alfa(N).
```

Quindi, spiegando chiaramente le ragioni, si indichino tutte le possibili risposte dell'interprete alla seguente query:

```
?- omega(X).
```