29 giugno 2010

Linguaggi di Programmazione

Nome e Cognome	
Corso di laurea	

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

```
a, b: integer;
s: string;
alfa: record (x: integer, y: string, beta: record (z: integer, w: integer));
v1: vector [5] of integer;
v2: vector [10] of vector [4] of integer;
a = 1;
s = "buongiorno";
b = a + 10;
alfa.x = b + 2 - a;
v1[a - v1[3]] = alfa.beta.z + (4 - v1[a-b]);
v2[7][3] = a + b - 2;
```

Tutte le dichiarazioni (se esistono) devono precedere gli assegnamenti (se esistono). Le variabili possono essere di tipo intero, stringa, record o vettore. I costruttori di tipo sono ortogonali fra loro. Negli assegnamenti, la parte sinistra può anche essere il campo di un record o l'elemento di un vettore. Le espression (di indicizzazione e di assegnamento) possono coinvolgere le operazioni aritmetiche di somma e differenza.

2. E' dato il seguente frammento di grammatica BNF relativo alla specifica di una istruzione condizionale in un linguaggio imperativo:

if-stat → **if** *expr* **then** *stat elsif-part* **else** *stat* **endif**
elsif-part → **elsif** *expr* **then** *stat elsif-part* | **ε**

Si chiede di specificare la semantica denotazionale del corrispondente frammento di linguaggio assumendo la disponibilità delle seguenti funzioni ausiliarie (di cui non è richiesta la specifica):

- M_{expr}(*expr*, s): restituisce il valore logico di *expr* allo stato s oppure ERRORE;
- M_{stat}(*stat*, s): restituisce il nuovo stato oppure ERRORE dopo l'esecuzione di *stat*.

In particolare, si richiede la specifica della seguente funzione semantica:

$M_{if\text{-}stat}$(*if-stat*, s).

3. Codificare nel linguaggio *Scheme* la funzione **`somma2`** che, ricevendo in ingresso una lista **`numeri`**, genera la lista delle somme delle coppie di numeri consecutivi (con in coda l'eventuale numero spaiato), come nei seguenti esempi:

`numeri`	**`(somma2 numeri)`**
`()`	`()`
`(2)`	`(2)`
`(2 3)`	`(5)`
`(2 3 4 5)`	`(5 9)`
`(2 3 4 5 1)`	`(5 9 1)`
`(2 3 4 5 1 6 7)`	`(5 9 7 7)`

4. Definire nel linguaggio *Haskell* una tabella in cui ogni riga associa al nome di uno studente la lista dei suoi esami. Ogni esame è rappresentato dal nome del corso e dal voto conseguito. Quindi, codificare la funzione **`superato`** (protocollo incluso) che, ricevendo in ingresso un **`esame`** (nome del corso) e la tabella degli studenti con i relativi esami, genera la lista dei nomi degli studenti che hanno superto l'**`esame`**.

5. È data la base di fatti *Prolog* che rappresenta un albero. Ogni fatto specifica un frammento di albero, cioè il collegamento di un nodo padre con i suoi figli, come nel seguente esempio (in cui `p` è il padre e `f1`, `f2`, `f3` i figli):

```
frammento(p,[f1, f2, f3]).
```

Si chiede di specificare il predicato **`equinodi(N1,N2)`**, che risulta vero qualora `N1` ed `N2` siano due nodi (diversi fra loro) posizionati allo stesso livello nell'albero.

6. Mediante l'ausilio di un semplice esempio, illustrare le scelte progettuali relative all'ambiente di referenziazione di sottoprogrammi passati come parametri.