25 luglio 2005

Linguaggi di Programmazione

Nome e Cognome	
Matricola	
Anno di corso	
Telefono	

1. Specificare la EBNF di un linguaggio per definire *ambienti computazionali*, come nel seguente esempio:

```
env alfa is
    a, b: integer;
    c: string;
    function beta(x: real, y: matrix[2,3] of string): real;
    function gamma(z: string, w: matrix[2] of matrix[5,6,7] of integer): matrix[10] of real;
    d, e, f: matrix[2,20] of integer;
    env delta is
        function epsilon(t: string): integer;
        g: real;
        h: string;
    end;
    k: integer;
    function zeta(): string;
    env lambda is
        m: matrix[10,20] of integer;
    end;
    function omega(s: string): string;
end;
```

Un ambiente è definito da un nome (racchiuso dalle keyword **`env`** ed **`is`**) e termina con la keyword **`end`**. Il corpo dell'ambiente è costituito da una lista non vuota di dichiarazioni che possono essere liste di variabili, funzioni o ambienti. Non esiste un ordine predefinito per tali dichiarazioni. Una lista di variabili è definita dal nome delle variabili seguite dal tipo. Un tipo può essere semplice (**`integer`**, **`real`**, **`string`**) o matriciale (**`matrix`**). Un tipo matriciale specifica una matrice di $n \geq 1$ dimensioni, in cui ogni dimensione è definita da un intero positivo. Una funzione è definita da un nome, dalla lista (eventualmente vuota) di parametri e dal tipo del valore di ritorno.

2. È data la seguente tabella di operatori per la quale si assume priorità decrescente dall'alto verso il basso, valutazione degli operandi da sinistra a destra e valutazione delle espressioni logiche in corto circuito:

Operatori	*Associatività*
`*`	sinistra
`+`	sinistra
`==`	nonassoc
`and`	sinistra
`or`	sinistra

e la seguente istruzione di assegnamento:

```
x := a + b + c * d * f == g * h + i or j == k and m + n == p
```

a) Rappresentare l'albero della espressione di assegnamento.
b) Specificare la semantica operazionale dell'istruzione di assegnamento.

NB: Il linguaggio di specifica operazionale è così caratterizzato:

- Contiene gli operatori **+**, *****, **==**;
- Non contiene gli operatori logici **and**, **or**;
- Contiene l'operatore logico **not**;
- Ogni operatore non può essere applicato ad espressioni, ma solo a variabili;
- Contiene le istruzioni condizionali (*if-then* ed *if-then-else*) i cui predicati sono variabili.

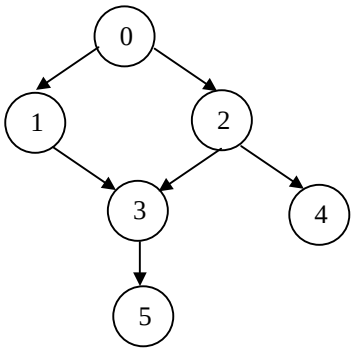
3. Definire nel linguaggio *Haskell* le seguenti strutture tabellari:

```
Citta(NomeCitta, NumAbitanti)
Fiumi(NomeFiume, Lunghezza)
Attraversamenti(NomeFiume, NomeCitta)
```

Quindi, codificare la funzione `grosse_citta_con_lunghi_fiumi` che, ricevendo in ingresso le tabelle `citta`, `fiumi` ed `attraversamenti`, computa la lista delle città che hanno più di 1.000.000 di abitanti e sono attraversate da un fiume lungo più di 500 chilometri.

4. È data una base di fatti *Prolog* relativa alla specifica di un grafo connesso, aciclico e diretto, come nel seguente esempio:

```
node(0, [1,2]).
node(1, [3]).
node(2, [3,4]).
node(3, [5]).
node(4, []).
node(5, []).
```



Ogni nodo è rappresentato dal suo identificatore e dalla lista di nodi successivi. Il grafo stabilisce un ordinamento temporale parziale tra eventi. Ad esempio, l'evento 0 accade prima di ogni altro evento. Gli eventi 1 e 2 accadono dopo l'evento 0. L'evento 3 accade dopo gli eventi 1 e 2. L'evento 1 accade prima dell'evento 5. D'altra parte, alcuni eventi non sono confrontabili. Ad esempio, non è possibile stabilire una relazione d'ordine tra gli eventi 1 e 2, nemmeno tra 4 e 5. Per definizione, ogni evento è confrontabile con se stesso. In sintesi, due eventi possono essere o meno confrontabili.

Sulla base di tale interpretazione del grafo, si chiede di specificare la seguente regola:

```
confrontabili(N1, N2)
```

in cui gli eventi `N1` ed `N2` sono confrontabili.

5. Illustrare il concetto di conversione di tipo implicita, con relativi vantaggi e svantaggi.

6. Elencare le possibili scelte progettuali relative alla definizione di sottoprogrammi.