15 luglio 2008

Linguaggi di Programmazione

Nome e Cognome	
Corso di laurea	
Telefono	
Email	

1. Specificare la grammatica EBNF di un linguaggio di interrogazione SQL-like, in cui ogni frase comprende una o più interrogazioni, come nel seguente esempio:

```
SELECT voto
FROM Esami;

SELECT nome, cognome
FROM Persone
WHERE citta = 'milano';

SELECT AVG(voto) AS media
FROM Esami
WHERE matricola = 1246;

SELECT Studenti.matricola AS codice, AVG(Esami.voto)
FROM Studenti, Esami
WHERE Studenti.matricola = Esami.matricola AND
      Studenti.matricola > 1000 AND Studenti.matricola < 2000
GROUP BY Studenti.matricola;

SELECT nome, cognome, MIN(voto), AVG(voto), MAX(voto)
FROM Studenti, Esami
WHERE Studenti.matricola = Esami.matricola AND
      Studenti.matricola > 1000 AND Studenti.matricola < 2000
GROUP BY Studenti.matricola
HAVING COUNT(*) > 10 AND MIN(voto) > 20
ORDER BY cognome, nome;
```

In ogni interrogazione, le clausole **SELECT** e **FROM** sono obbligatorie, mentre le clausole **WHERE**, **GROUP BY**, **HAVING** e **ORDER BY** sono facoltative. Inoltre, la clausola **HAVING** può essere specificata solo se è stata specificata la clausola **GROUP BY**. La clausola **SELECT** specifica una lista di attributi di relazione e/o funzioni aggregate su attributi. Possibili funzioni aggregate sono **COUNT**, **MIN**, **MAX** ed **AVG**. La funzione **COUNT** ha sempre come argomento un asterisco, mentre le altre funzioni hanno come argomento un attributo di relazione. Il nome di un attributo può essere preceduto dal nome della relazione di cui fa parte. Ogni elemento listato nella clausola **SELECT** può essere ridenominato mediante la keyword **AS** seguita da un identificatore. La clausola **FROM** specifica una lista di nomi di relazioni. La clausola **WHERE** specifica il predicato di selezione, composto in generale da una catena di congiunzioni di condizioni. Una condizione è il confronto tra un attributo ed una costante (intera o stringa) o un altro attributo. Possibili confronti sono **=**, **<** , **>**. La clausola **GROUP BY** specifica una lista di attributi di raggruppamento. La clausola **HAVING** specifica un predicato sui gruppi di tuple stabiliti dalla clausola **GROUP BY**. Le condizioni espresse dalla clausola **HAVING** sono rappresentate da un confronto tra una funzione aggregata ed una costante intera o un'altra funzione aggregata. Infine, la clausola **ORDER BY** specifica una lista di attributi sui quali ordinare le tuple risultanti. Ogni interrogazione termina con un punto e virgola.

2. Specificare la semantica denotazionale della seguente istruzione (ciclo *for C-like*):

for (I_1; B; I_2) L

la cui semantica operazionale può essere espressa mediante il ciclo *while* nel seguente modo:

I_1;
while B **do** L; I_2 **end.**

dove I_1 ed I_2 sono istruzioni, B è una esprressione booleana ed L è una lista di istruzioni (corpo del ciclo). Si assume che siano disponibili le seguenti funzioni semantiche (di cui non è richiesta l'implementazione):

- $\mathbf{M_{stat}(I, s)}$, che computa il nuovo stato (eventualmente ERRORE) dopo l'esecuzione della istruzione **I** nello stato **s**;
- $\mathbf{M_{bool}(B, s)}$, che computa il valore dell'espressione booleana **B** (eventualmente ERRORE) nello stato **s**;
- $\mathbf{M_{list}(L, s)}$, che computa il nuovo stato (eventualmente ERRORE) dopo l'esecuzione della lista **L** di istruzioni nello stato **s**.

3. Specificare nel linguaggio *Scheme* la funzione unaria **`duplica`** che, ricevendo in ingresso una **`lista`**, computa una nuova lista, in cui ogni elemento di **`lista`** viene duplicato in una coppia di elementi uguali. Ecco alcuni esempi:

`lista`	**`(duplica lista)`**
`()`	`()`
`(a)`	`((a a))`
`(a b c)`	`((a a)(b b)(c c))`
`(1 () (2 3) (4))`	`((1 1)(())(())((2 3)(2 3))((4)(4)))`

4. Definire nel linguaggio *Haskell*, mediante la notazione di pattern-matching, la funzione **`applica`** (protocollo incluso), avente in ingresso due funzioni ed una lista di interi. Le due funzioni mappano un intero in un altro intero. La funzione **`applica`** genera la lista di interi in cui ogni elemento rappresenta il valore della applicazione della prima funzione al risultato della seconda funzione applicata ad un elemento della lista in ingresso. Ecco alcuni esempi:

- `applica quadrato cubo [1,2,3] = [1,64,729]`
- `applica doppio fattoriale [1,3,5] = [2,12,240]`
- `applica fattoriale doppio [1,3,5] = [2,720,3628800]`

5. E' data una base di fatti *Prolog* relativa ad un albero genealogico, come nel seguente esempio:

```
capostipite(vincenzo).
genitore(vincenzo,[luisa,franco,elena]).
genitore(luisa,[andrea,dario,guido]).
genitore(franco,[giovanni,paola,letizia,sofia]).
genitore(elena,[francesco,angelo]).
...
```

in cui **`capostipite`** indica il capostipite dell'albero genealogico, mentre **`genitore`** associa ad ogni elemento dell'albero una lista (eventualmente vuota) di figli.
Si chiede di specificare il predicato **`stessa_generazione(X,Y)`**, che risulta vero qualora le persone **`X`** ed **`Y`** siano della stessa generazione (e diverse fra loro), cioè, allo stesso livello nell'albero genealogico (ad esempio, **`francesco`** e **`giovanni`**, ma non **`francesco`** e **`luisa`**).

6. Illustrare i principi della semantica assiomatica.