15 Luglio 2015

Linguaggi di Programmazione

Cognome e nome	
Matricola	
Registrazione esame integrato?	**Sì**: ☐ **No**: ☐

1. Specificare la definizione regolare relativa ad una tabella (anche vuota), in cui ogni riga contiene i dati di una persona descritta da nome (anche più di uno), cognome (anche più di uno), via, numero civico (massimo 3 cifre, non inizia con 0), città (composta eventualmente da più nomi) e provincia (identificata dalla sigla tra parentesi), come nel seguente esempio:

```
Andrea Lorenzi, Via Italia 135, Urago Mincio (PV)
Rita Sandra De Benedetti, Via Alberelle 8, Panoramico (BG)
Angelo Rinaldi, Via Sentieri 18, Monte Alto Toscano (GR)
```

È richiesto che la specifica sia conforme alla spaziatura e alla punteggiatura indicata nell'esempio.

2. Specificare la grammatica BNF di un linguaggio per la definizione e manipolazione di array, come nel seguente esempio:

```
a, b, c: array [100] of integer;
s: array [10] of string;
alfa: array [20] of array [100] of integer;
a = b;
a[2] = b[3];
alfa[3] = a;
alfa[15][58] = 2;
s[3] = "beta";
b = alfa[8];
```

La frase può anche essere vuota. Ci sono due tipi di istruzioni: definizione di array ed assegnamento. Nell'assegnamento, la parte sinistra può solo essere un array o una espressione di indicizzazione (eventualmente con indici multipli). La parte destra può anche essere una costante intera o stringa.

3. Specificare in un linguaggio imperativo la semantica operazionale del seguente frammento di codice *Haskell:*

```
type Matricola = Integer
type Anno = Integer
type Corso = String
type Nome = String

type Studenti = [(Matricola, Anno, Corso)]
type Docenti = [(Nome, Corso)]

docentiStudente :: Studenti -> Docenti -> Matricola -> [ Nome ]
docentiStudente stud doc mat =
   [ n | (m, a, c) <- stud, (n, c') <- doc, c==c',m==mat ]
```

4. È dato il seguente frammento di grammatica BNF, relativo alla specifica del ciclo *foreach* in un linguaggio imperativo:

foreach-stat → **foreach id in id do** *stat* **endfor**

```
foreach num in numeri do
  print(num)
endfor;
```

esempio di frase

Specificare la semantica denotazionale del frammento di grammatica, assumendo che siano disponibili le funzioni ausiliarie M_{id}(**id**, `s`), che restituisce il valore di **id** allo stato `s` (eventualmente **errore**), M_{assign}(**id**, val, `s`), corrispondente all'assegnamento nello stato `s` della variabile **id** mediante il valore `val`, e M_{stat}(*stat*, `s`).
In particolare, specificare la funzione semantica $M_{foreach}$(*foreach-stat*, `s`).

5. Specificare in *Scheme* la funzione `treccia`, avente in ingresso due liste, `A` e `B`, che computa l'intreccio delle due liste, come nei seguenti esempi:

```
(treccia '(a b c) '(1 2 3)) = (a 1 b 2 c 3)
(treccia '(a b c) '(1 2 3 4 5)) = (a 1 b 2 c 3 4 5)
(treccia '(a b c d) '(1 2)) = (a 1 b 2 c d)
(treccia '() '(1 2 3)) = (1 2 3)
(treccia '(a b c) '()) = (a b c)
(treccia '() '()) = ()
```

Si noti che, nel caso di diversa lunghezza delle liste, gli elementi in eccesso vengono accodati nel risultato.

6. È data una struttura tabellare che rappresenta una relazione di amicizie, come nel seguente esempio:

Nome	Amici
luigi	*angelo* *rina*
anna	*maria* *anna* *stefano* *carlo*
letizia	*camilla* *andrea* *giuseppe*
paola	*giovanni* *martina* *paola*

Specificare in *Haskell* la funzione `filtra` che, ricevendo in ingresso una lista di amicizie (come nell'esempio), computa la lista delle persone che hanno più di tre amici ed uno di questi omonimo (nell'esempio, ["*anna*"]).

7. Si vuole fare una fotografia di cinque amici, *maria*, *giovanni*, *marta*, *matteo* e *linda*, disposti da sinistra a destra. Specificare in *Prolog* il predicato `fotografia(A1, A2, A3, A4, A5)`, in cui `Ai` è il nome dell'amico nella posizione `i`-esima, che risulta vero qualora siano rispettati i seguenti vincoli: *marta* e *matteo* non sono in terza posizione, *giovanni* non è ne in prima ne in ultima posizione e si trova tra *marta* (immediatamente a sinistra) e *matteo* (immediatamente a destra), *maria* si trova in una posizione a sinistra di *giovanni*, *linda* si trova immediatamente a destra di *matteo*.

8. Dopo aver descritto la funzione `eval` nel linguaggio *Scheme*, illustrare l'interpretazione della seguente espressione, precisando gli argomenti intermedi delle varie chiamate ed il risultato finale:

```
(eval (eval ''(car (a b c d))))
```