Linguaggi di Programmazione

Nome e cognome	
Anno di corso	

1. Specificare la grammatica BNF di un linguaggio in cui ogni frase corrisponde ad una o più dichiarazioni di variabili. Ogni dichiarazione è terminata dal separatore '**;**'. Ogni variabile può essere di tipo semplice (**`int`**, **`string`**, **`bool`**) o di tipo record (lista di uno o più campi racchiusa tra parentesi, ognuno dei quali può essere a sua volta semplice o complesso), come nella seguente frase:

```
x: string;
alfa: (omega: string);
delta: (A: string, B: int, C: bool);
z: (D: int, E: (beta: string, gamma: (F: bool, G: int)), H: int);
```

   Se il record include più di un campo, questi devono essere separati da una virgola.

   NB: Non è richiesta la specifica degli identificatori (considerati terminali).

2. Esprimere la grammatica BNF relativa al punto 1 mediante la notazione *Definite Clause Grammar* del linguaggio *Prolog*.

3. Definire nel linguaggio *Scheme* la funzione `atomi`, avente in ingresso una `lista`, che computa la lista degli elementi atomici (non liste) di `lista`. Ecco alcuni esempi:

`lista`	`atomi`
`(()())`	`()`
`(A (B C))`	`(A B C)`
`((A B) C (D (E F) G))`	`(A B C D E F G)`

4. Definire in *Smalltalk* l'espressione di messaggio in cui il ricevente `W` è un vettore (non vuoto) di vettori (non vuoti) di interi. Il messaggio deve filtrare (funzionalmente) gli elementi di `W` il cui ultimo elemento è negativo, come nei seguenti esempi:

`W`	*Risultato*
`#(#(1 2 -3) #(4 5) #(6 -7))`	`#(#(1 2 -3) #(6 -7))`
`#(#(1 3) #(4 5) #(6 7 8))`	`#()`
`#(#(1 -3) #(4 -5) #(6 -8))`	`#(#(1 -3) #(4 -5) #(6 -8))`

5. Discutere la correlazione tra i requisiti di qualità del software e le caratteristiche dei linguaggi di programmazione che le supportano.

6. Enunciare e giustificare informalmente le regole di overriding dei metodi (nel paradigma ad oggetti) che garantiscono che una sottoclasse sia anche un sottotipo.