

Linguaggi di Programmazione

Nome e Cognome	
Corso di laurea	
Telefono	
Email	

1. Specificare la EBNF di un linguaggio imperativo in cui ogni programma è composto da una intestazione, da una sezione (opzionale) di dichiarazione di variabili e da un corpo, come nel seguente esempio:

```

program primavera
  var a, b1, beta, i: int;
      c, z: real;
      d, e, omega25: string;
  begin
    a := 10;
    b1 := a;
    if a > 3 then
      c := 3.0;
      z := 12.24;
    elsif a < b then
      z := a;
    elsif a = b then
      d := "alfa";
      for i := 1 to 25 do
        if c > z then
          omega25 := e;
        endif;
      endfor;
    otherwise
      while a != i do
        a := b1;
        b1 := 3;
      endwhile;
    endif;
  end.

```

----- intestazione del programma

----- dichiarazione di variabili

----- corpo del programma

Le variabili possono essere di tipo **int**, **real** o **string**. Il corpo del programma è costituito da una sequenza non vuota di istruzioni racchiusa tra **begin** ed **end**. Ogni istruzione può essere un assegnamento, una istruzione condizionale a più vie, un ciclo a condizione iniziale o un ciclo a conteggio. Possono essere assegnate variabili con altri variabili o valori. Una istruzione condizionale **if ... endif** coinvolge zero o più rami **elsif** e, opzionalmente, il ramo **otherwise**. Una condizione è il confronto (**=**, **!=**, **>**, **<**, **>=**, **<=**) tra una variabile e un valore o un'altra variabile. Ad ogni iterazione del ciclo a conteggio **for ... endfor**, la variabile di conteggio può incrementare (**to**) o decrementare (**downto**). Il corpo dei cicli (**for** e **while**), del **then** e dell'**otherwise** è costituito da una lista non vuota di istruzioni.

2. Specificare la semantica denotazionale di una espressione logica definita dalla seguente grammatica:

$$expr \rightarrow \text{true} \mid \text{false} \mid \text{id} \mid expr_1 \text{ and } expr_2 \mid expr_1 \text{ or } expr_2$$

sulla base delle seguenti assunzioni:

- id** rappresenta il nome di una variabile logica;
 - la valutazione degli operatori **and** ed **or** è in corto circuito;
 - la valutazione degli operandi dell'operatore **and** è da sinistra a destra;
 - la valutazione degli operandi dell'operatore **or** è da destra a sinistra;
 - è disponibile una funzione $\mu(\text{id}, s)$ che restituisce il valore della variabile **id** nello stato s ;
 - $\mu(\text{id}, s) = \text{UNDEF}$, qualora il valore della variabile **id** non sia definito;
 - il linguaggio di specifica denotazionale non dispone di operatori logici.
3. Definire nel linguaggio funzionale *Scheme* la funzione `shrink`, avente in ingresso una lista L , che computa la lista degli elementi di L che si trovano in posizione dispari, come nei seguenti esempi

L	$(\text{shrink } L)$
$()$	$()$
(a)	(a)
$(a \ b)$	(a)
$(a \ b \ c)$	$(a \ c)$
$(a \ (1 \ 2 \ 3) \ (4 \ 5 \ (6 \ 7)) \ 8 \ ())$	$(a \ (4 \ 5 \ (6 \ 7)) \ ())$

4. Definire nel linguaggio *Haskell* la funzione `shrink` al punto 3 mediante pattern matching, sulla base del seguente protocollo: `shrink :: [a] -> [a]`.

5. Definire nel linguaggio logico *Prolog* il predicato `shrunked(L, S)`, che risulta vero quando S rappresenta il risultato della funzione `shrink(L)` specificata al punto 3.

6. Discutere le principali scelte progettuali relative alla definizione di un linguaggio orientato agli oggetti.