Linguaggi di Programmazione

Nome e Cognome	
Corso di laurea	

1. Specificare la grammatica BNF di un linguaggio per la dichiarazione di variabili, come nel seguente esempio:

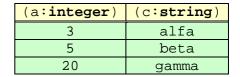
```
a, x1, gamma: integer;
s: string;
b1, b2: boolean;
v: vector(3,5,10) of string;
s: structure(a: integer, b: vector(2,5) of string);
f, g: function() -> vector(10) of integer;
omega: function(integer, structure(codice: string, prezzo: integer)) -> boolean;
```

Ogni frase specifica una lista (non vuota) di dichiarazioni. Partendo dai tipi elementari integer, string e boolean, si possono specificare espressioni di tipo mediante i costruttori vector, structure e function. Un vettore viene qualificato da una lista (non vuota) di dimensioni. Una struttura è definita da una lista (non vuota) di attributi. Una funzione è definita dal suo protocollo (parametri anonimi). Il linguaggio non è ortogonale poichè il tipo di un vettore è sempre atomico e una funzione non è una forma funzionale (non può ricevere o restituire funzioni).

2. Specificare la semantica operazionale dell'operatore relazionale di proiezione mediante una notazione imperativa:

in cui X è l'operando della proiezione ed A la lista degli attributi di X su cui effettuare la proiezione. Ecco un esempio (in cui la parte in giallo è lo schema, mentre la parte in verde è l'istanza):

(a:integer)	(b:bool)	(c:string)	(d:integer)
3	true	alfa	23
5	false	beta	12
20	true	gamma	5
3	false	alfa	8



In particolare, si richiede di computare le variabili sy ed iy, che rappresentano rispettivamente lo schema e l'istanza del risultato. Si richiede inoltre di verificare che la lista A non contenga nomi che non siano attributi dello schema dell'operando. Si assumono le seguenti funzioni ausiliarie (di cui non è richiesta la codifica):

- schema(X): lista di coppie (nome, tipo) che definiscono lo schema di X;
- instance(X): lista di tuple che definiscono l'istanza di X;
- attributes(A): lista degli attributi di proiezione in A;
- **error**(): funzione di errore (chiamata in caso di errore semantico).
- 3. Definire nel linguaggio *Scheme* la funzione **valori**, la quale, ricevendo in ingresso un numero naturale \mathbf{n} ed una funzione unaria \mathbf{f} di numeri naturali, computa la lista dei valori $\mathbf{f}(0)$, $\mathbf{f}(1)$, ..., $\mathbf{f}(n)$.

4. Definire nel linguaggio *Haskell* la funzione **affetta** (protocollo incluso), la quale, avente in ingresso un numero intero **n**>0 ed una **lista**, genera la lista di liste ottenuta prelevando ripetutamente **n** elementi da **lista**, come nei seguenti esempi:

n	lista	(affetta n lista)
2	[1,2,3,4,5,6,7,8,9,10]	[[1,2],[3,4],[5,6],[7,8],[9,10]]
3	[True,False,True,False,True]	[[True,False,True],[False,True]]
1	"alfabeto"	["a","l","f","a","b","e","t","o"]

5. E' dato un fatto *Prolog* che specifica una lista di numeri interi, come nel seguente esempio:

```
numeri([-10,-9,-8,-7,-6,-5,-4,-3,-2,-1,0,1,2,3,4,5,6,7,8,9,10]).
```

Si chiede di specificare il predicato **scomponi** (A,B,C,D), che risulta vero qualora A e B siano i coefficienti di una equazione di secondo grado \mathbf{x}^2 +Ax +B = 0 che possa espressa come prodotto $(\mathbf{x}+\mathbf{C})(\mathbf{x}+\mathbf{D}) = 0$, in cui C e D appartengono alla lista argomento del fatto **numeri**. Si ricorda che, ai fini della scomposizione, A e B devono corrispondere rispettivamente alla somma ed al prodotto di C e D. Ad esempio, \mathbf{x}^2 +3x -4 = 0 può essere espressa come $(\mathbf{x}-1)(\mathbf{x}+4) = 0$, quindi:

```
?- scomponi(3,-4,C,D).
C = -1
D = 4
```

Si richiede anche che la regola sia specificata in modo tale che l'interprete dia un'unica soluzione (nel nostro esempio, la seconda soluzione (simmetrica, ma che non deve essere fornita) sarebbe C = 4, D = -1).

- **6.** Discutere l'interpretazione dei seguenti cinque goal *Prolog*, indicando per ognuno di essi la risposta dell'interprete:
 - \bullet ?- X = Y.
 - ?- X == Y.
 - ?- X = 4+6, X = Y.
 - ?-X = Y, X == Y, X = 1.
 - ?- 10 **is** X+6.