9 settembre 2003 Durata: 2 ore

Linguaggi di Programmazione

Nome e Cognome	
Matricola	
Anno di corso	
Telefono	

1. Specificare la grammatica EBNF di un linguaggio in cui ogni frase corrisponde ad una o più dichiarazioni di variabili. Ogni dichiarazione è definita dal nome della variabile e dal suo tipo. Possibili tipi sono **`int`**, **`string`**, **`bool`** e **`table`**. A differenza dei primi tre, che sono semplici, il tipo **`table`** definisce una tabella le cui colonne sono a loro volta caratterizzate da identificatori e relativi tipi, incluso **`table`**. In questo modo, il linguaggio permette la definizione di tabelle complesse (tabelle che incorporano tabelle, senza limiti di profondità). Ecco un esempio di frase (si noti che la descrizione di una tabella è racchiusi tra parentesi graffe):

```
alfa: int
beta: string
T: table {A: string, B: table {C: bool, D: int}, E: string}
```

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

2. Data la seguente tabella di descrizione di operatori aritmetici (precedenza decrescente dall'alto verso il basso):

Operatori	Associatività
`^`	destra
`*, /`	sinistra
`+, -`	destra

   riscrivere la seguente espressione in forma parentetica:

```
a + b - c * d + e ^ f ^ g + h / i / l + m * n
```

3. Supponendo la modalità di passaggio dei parametri per nome, determinare il valore del vettore `vet` dopo l'esecuzione della funzione `beta` nel seguente frammento di codice:

```
int vet[4]= {3,3,3,3}, x = 2;
void alfa(int a, b)
{
    a = 5;
    b = b + 1;
    a = 4;
}
void beta()
{
    alfa(vet[x], x);
}
```

   NB: Si assume che il primo elemento del vettore sia `vet[0]`.

9 settembre 2003 Durata: 2 ore

4. Definire nel linguaggio *Haskell* una struttura tabellare

```
Esami (Matricola, Corso, Voto)
```

in cui ogni riga rappresenta rispettivamente uno studente, il corso, ed il relativo voto. Quindi, codificare in Haskell la funzione `esamiSostenuti` che, ricevendo in ingresso una tabella `esami` ed una certa `matricola`, computa la lista degli esami superati dal relativo studente (senza l'indicazione del voto).

5. Elencare i diversi tipi di binding-time con il supporto di semplici esempi.

6. Illustrare le possibili scelte progettuali per linguaggi di programmazione orientati agli oggetti.