PROGRAMMAZIONE 1 e LABORATORIO (A,B) - a.a. 2008-2009
Prova scritta del 15 giugno 2009

Scrivere **IN STAMPATELLO** COGNOME, NOME e CORSO su ogni foglio consegnato

ESERCIZIO 1 (punti 6)
Data la seguente grammatica libera sull'alfabeto $\Lambda = \{a, b\}$

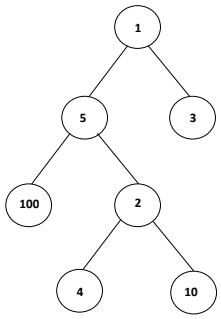
$$S \longrightarrow SS \mid aSb \mid ab$$

dimostrare che è ambigua e definire una grammatica libera **non ambigua** che genera lo stesso linguaggio.

ESERCIZIO 2 (punti 6)
Dato il tipo degli alberi binari visto a lezione

```
type 'a btree = Empty  | Node of 'a * 'a btree * 'a btree
```

si definisca una funzione `leaves : 'a btree -> 'a list` in modo che `(leaves bt)` sia la lista costituita da tutti gli elementi contenuti nelle foglie di `bt`, presi da sinistra a destra. Ad esempio, dato l'albero `alb` rappresentato graficamente in figura



la chiamata `(leaves alb)` deve restituire la lista `[100; 4; 10; 3]`.

ESERCIZIO 3 (punti 6)
Si definisca in `C` una procedura

```
void check (int  vet[], int dim, int size, int *esito)
```

che, dato un array `vet` di dimensione `dim` e un intero positivo `size`, controlla che la sequenza degli elementi massimi nelle porzioni dell'array `vet` con indici nell'intervallo `[i*size, i*size+size)`, con `i` crescente nell'intervallo `[0, dim/size)`, sia crescente. La procedura deve lasciare il risultato della verifica nella variabile puntata da `esito`. Ad esempio, dato l'array `v` in figura

12	1	15	33	3	2	44	25	6	18

la chiamata `check(v,10,3)` deve lasciare in `*esito` il valore `true` (essendo crescente la sequenza `15 33 44`) mentre la chiamata `check(v,10,2)` deve lasciare in `*esito` il valore `false` (non essendo crescente la sequenza `12 33 3 44 18`).

ESERCIZIO 4 (punti 6)
Si supponga di estendere la sintassi dei comandi con la nuova produzione

```
Com ::= *Ide-- if positive
```

Informalmente, `*x-- if positive` ha l'effetto di decrementare di 1 il valore della variabile puntata da `x` se tale valore è maggiore di 0, di lasciarlo inalterato altrimenti. Dare la semantica formale del nuovo comando.

ESERCIZIO 5 (punti 6)
Senza utilizzare ricorsione esplicita, ma utilizzando funzioni di ordine superiore, si definisca una funzione `interleave` con tipo

```
interleave: 'a list -> 'a list -> 'a list
```

in modo che `(interleave l1 l2)` sia la lista ottenuta inserendo `l2` dopo ogni elemento di `l1`. Ad esempio la chiamata `interleave [1;2;3;4] [0;0;0]` deve restituire la lista `[1;0;0;0;2;0;0;0;3;0;0;0;4;0;0;0]`.