PROGRAMMAZIONE 1 e LABORATORIO (A,B)
a.a. 2010-2011

Prova scritta del 15 febbraio 2011

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

ESERCIZIO 1 (punti 5)
Dato l'alfabeto $\Lambda = \{1, 2, 3\}$, si definisca una grammatica libera che genera il seguente linguaggio $\mathcal{L} \subseteq \Lambda^*$

$$\mathcal{L} = \{\alpha\beta \mid \alpha, \beta \in \Lambda^+, \alpha \text{ non decrescente}, \beta \text{ non crescente}\}$$

Si ricorda che $\Lambda^+ = \Lambda^* \setminus \{\epsilon\}$.

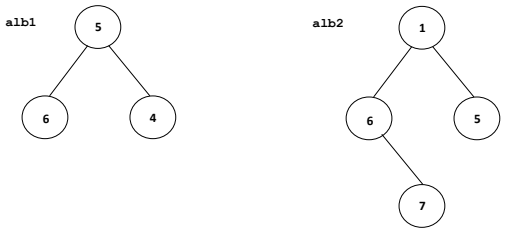
ESERCIZIO 2 (punti 5)
Dato il tipo degli alberi binari visto a lezione

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

si definisca una funzione

```
valdepth : int btree -> int * int
```

che restituisce il valore e la profondità della foglia più a destra di profondità massima. Ad esempio, dati i due alberi `alb1` e `alb2` in figura



`valdepth alb1 = (4,2)` e `valdepth alb2 = (7, 3)`. Per semplicità, si supponga `valdepth Void = (0, 0)`.

ESERCIZIO 3 (punti 5)
Scrivere una funzione `C` che, dato un array a di dimensione dim, restituisce $true$ se e solo se è verificata la seguente formula

$$\forall i \in [0, dim-1).\big((\exists j \in [i+1, dim).a[i] > a[j]) \wedge (\forall k \in [i+1, dim).a[i] \geq a[k])\big)$$

ESERCIZIO 4 (punti 5)
Si definisca in `C` una procedura che, dati due array a e b di dimensioni $dima$ e $dimb$ rispettivamente, azzera tutti gli elementi di a che non compaiono in b.

ESERCIZIO 5 (punti 5)
Senza utilizzare ricorsione esplicita, ma solo funzioni di ordine superiore, si definisca in CAML una funzione

```
DelOutside : (int * int) list -> (int * int) ->  (int * int) list
```

che, presa una lista di coordinate intere di punti nel piano cartesiano e le coordinate di un punto `(x, y)` nel primo quadrante (ovvero con `x` ≥ 0 e `y` ≥ 0), cancella dalla lista tutte le coordinate di punti che non stanno nel rettangolo con vertici nei punti `(0,0)`, `(x,0)`, `(0,y)` e `(x,y)`.

ESERCIZIO 6 (punti 5)
Si suppongano date le seguenti definizioni:

```
struct el { T info; struct el *next;};
typedef struct el ElementoLista;
typedef ElementoLista *ListaDiElementi;
```

dove `T` è un generico tipo. Scrivere in `C` una procedura che cambia l'ordine degli ultimi due elementi della lista (se esistono). La procedura deve cambiare l'ordine delle due ultime strutture di tipo `ElementoLista` e non può agire mediante assegnamenti sui valori dei rispettivi campi `info`.