PROGRAMMAZIONE 1 e LABORATORIO (A,B) - a.a. 2013-2014
II Verifica scritta del 20/12/2013

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

ESERCIZIO 1 (6 punti)
Dato il tipo degli alberi binari

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

si definisca in `CAML` una funzione `maxunico` con tipo

```
maxunico : 'a btree -> 'a * bool
```

in modo che `(maxunico bt)` restituisca la coppia `(m, b)` dove `m` è l'elemento massimo di `bt` e `b` è `true` se `m` occorre in `bt` esattamente una volta, `false` altrimenti. La funzione non è definita su alberi vuoti.

ESERCIZIO 2 (6 punti)
Senza utilizzare ricorsione esplicita, definire in CAML una funzione

```
foo : int list -> int * int
```

in modo che `(foo lis)` restituisca:

- la coppia costituita dagli ultimi due elementi dispari di `lis`, se questa contiene almeno due elementi dispari
- la coppia `(0, d)` se `d` è l'unico elemento dispari di `lis`
- la coppia `(0, 0)` se `lis` non contiene elementi dispari.

ESERCIZIO 3 (6 punti)
Scrivere in `C` una funzione

```
int check (int a [], int b [], int dima, int dimb)
```

che, dati due array `a` e `b` di dimensione `dima` e `dimb` rispettivamente, calcola il valore di verità della seguente formula

$$(\forall i \in [0, dima).\exists j \in [0, dimb).a[i] = b[j]) \ \wedge \ (\forall i \in [0, dimb).\exists j \in [0, dima).a[j] = b[i])$$

ESERCIZIO 4 (6 punti)
Scrivere in `C` una procedura

```
void azzera (int a [], int b [], int dima, int dimb)
```

che, dati due array `a` e `b` di dimensione `dima` e `dimb` rispettivamente, azzera tutti gli elementi di `a` che occorrono in `b` più di una volta e lascia inalterati tutti gli altri elementi di `a`.

ESERCIZIO 5 (6 punti)
Date le seguenti definizioni:

```
struct el { int info; struct el *next;};

typedef struct el ElementoDiLista;
typedef ElementoDiLista *ListaDiInteri;
```

scrivere in `C` una procedura che, data in ingresso attraverso un opportuno parametro una lista di interi, elimina il penultimo elemento se è uguale all'ultimo, lascia la lista invariata in ogni altro caso.