PROGRAMMAZIONE 1 e LABORATORIO (A,B) - a.a. 2009-2010
Prova scritta del 30 giugno 2010

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

ESERCIZIO 1 (punti 5)
Si definisca una grammatica che genera il seguente linguaggio sull'alfabeto $\Lambda = \{a, b, c, d\}$

$$\mathcal{L} = \left\{a^n b^m c^k \mid pari(n+m) \wedge n, m, k > 0\right\} \cup \left\{a^n b^m d \mid dispari(n+m) \wedge n, m > 0\right\}$$

ESERCIZIO 2 (punti 5)
Si definisca in CAML una funzione `last_first` con tipo

```
last_first: 'a list list -> 'a list
```

che data una lista di liste `ll`, restituisce una lista costruita prendendo gli ultimi elementi (se esistono) di ciascuna lista in `ll`. Ad esempio

```
last_first [ [2;3] ; [ ]; [7;5;3] ; [4] ] = [3; 3; 4]
```

ESERCIZIO 3 (punti 5)
Si definisca in `C` una funzione

```
boolean check (int  a[], int dim)
```

che restituisce `true` se vale la seguente proprietà

$$\forall \texttt{i} \in [0, \texttt{dim}).\ \left(\exists \texttt{j} \in [0, \texttt{dim}).\ \texttt{j} \neq \texttt{i}\ \wedge\ \texttt{a}[\texttt{i}] = \texttt{a}[\texttt{j}]\right)$$

e restituisce `false` altrimenti (si supponga dato il tipo `typedef enum {false, true} boolean;`).

ESERCIZIO 4 (punti 5)
Senza utilizzare ricorsione esplicita, si definisca in CAML una funzione

```
sposta : int list -> int list
```

in modo che, se `foo l1 = l2`, la lista `l2` sia ottenuta da `l1` spostando tutti i valori negativi prima di quelli non negativi (non importa in quale ordine). Ad esempio:

```
sposta [-2; 3; 4; 5; -6; -7; 0] = [-2; -6; -7; 0; 5; 4; 3]
```

ESERCIZIO 5 (punti 5)
Supponiamo di rappresentare un insieme mediante una lista. Scrivere una procedura in `C` che, prese due liste rappresentanti due insiemi, modifica la prima in modo che rappresenti l'unione dei due insiemi. Si ricorda che in un insieme un elemento non può comparire più di una volta.

ESERCIZIO 6 (punti 5)
Scrivere in `C` una funzione `anagramma` che, dati due array di interi della stessa lunghezza e la loro dimensione, stabilisce se sono l'uno l'anagramma dell'altro (cioè contengono gli stessi valori con lo stesso numero di occorrenze). Ad esempio, i due array seguenti sono l'uno l'anagramma dell'altro

3	2	2	3	4

2	3	2	3	4

mentre non lo sono i due seguenti

3	3	3	2	4

3	2	2	3	4