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

Prova scritta del 21 luglio 2011

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

ESERCIZIO 1 (punti 5)
Data la seguente grammatica sull'alfabeto $\Lambda = \{$`0, 1`$\}$

```
S ::= 01 | 0S1 | SS
```

si dimostri che è ambigua e si definisca una grammatica non ambigua equivalente (che genera cioè lo stesso linguaggio).

ESERCIZIO 2 (punti 5)
Un albero *destrorso* è definito dal seguente tipo

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

Si definisca una funzione

```
rt2list : 'a rtree -> 'a list
```

in modo che `rt2list rt` restituisca la lista che contiene tutti gli elementi di `rt` in ordine crescente rispetto alla loro profondità.

ESERCIZIO 3 (punti 5)
Scrivere una funzione `C` che, dato un array di interi a di dimensione dim e un intero x, restituisce un indice i tale che

$$i \in [0, dim) \ \wedge \ \big(\sum_{j=0}^{i-1} a[j] < x < \sum_{j=0}^{i} a[j]\big)$$

se tale indice esiste, restituisce -1 altrimenti.

ESERCIZIO 4 (punti 5)
Un *multinsieme* può essere rappresentato come una lista di coppie in cui non compaiono mai due coppie con il primo elemento uguale. Ciascuna coppia rappresenta un elemento del multinsieme (primo elemento della coppia) ed il numero di volte che esso occorre nel multinsieme (secondo elemento della coppia).
Ad esempio, la lista `[(a,2); (z,1); (c,4); (d,2); (w,1)]` è una delle possibili rappresentazioni del multinsieme {`a,a,z,c,c,c,c,d,d,w`}.
Senza utilizzare ricorsione esplicita, ma solo funzioni di ordine superiore, si definisca in CAML una funzione con tipo

```
max_occ :  ('a * int) list -> 'a
```

che restituisce uno degli elementi che occorre il maggior numero di volte nel multinsieme rappresentato dal suo argomento.

ESERCIZIO 5 (punti 5)
Scrivere una funzione `C` che, dato un array a di dimensione dim, restituisce 1 se la sequenza $a[0]a[1]\ldots a[n-1]$ appartiene al linguaggio dell'esercizio 1, restituisce 0 altrimenti.

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

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

Scrivere in `C` una procedura che, data una lista, porta in testa il primo elemento maggiore dell'elemento immediatamente successivo e lascia la lista invariata se un tale elemento non esiste. **Nota bene**: la procedura non può utilizzare assegnamenti sui campi `info` degli elementi della lista.