PROGRAMMAZIONE 1 e LABORATORIO (A,B) - a.a. 2013/14

Prova scritta del 5 settembre 2014

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

ESERCIZIO 1 (punti 10)
Dato un alfabeto Λ e due stringhe $\alpha, \beta \in \Lambda^*$, si dice che α è sottostringa di β (che rappresentiamo con la notazione $\alpha \propto \beta$) se e solo se $\beta = \gamma\alpha\delta$ per qualche $\gamma, \delta \in \Lambda^*$.

1.1 (punti 5) Si definisca una grammatica per il seguente linguaggio sull'alfabeto $\Lambda = \{a, b, c\}$

$$\mathcal{L} = \{\alpha \mid \alpha \in \Lambda^+ \ \wedge \ aaa \not\propto \alpha\}$$

1.2 (punti 5) Si definisca un automa sull'alfabeto $\Lambda = \{a, b, c\}$ che riconosce il linguaggio

$$\mathcal{L} = \{\alpha \mid \alpha \in \Lambda^+ \ \wedge \ abc \not\propto \alpha\}$$

ESERCIZIO 2 (punti 5)
Il seguente tipo CAML puó essere utilizzato per la rappresentazione di *multinsiemi* di valori

```
type 'a multiset = Vuoto | Elemento of 'a * int * 'a multiset
```

La coppia di valori di tipo `'a * int` costituita dai primi due parametri di `Elemento` indica che il valore di tipo `'a` compare nel multinsieme tante volte quante indicate dal valore di tipo `int`. Ad esempio il multinsieme di interi $\{100, 2, 100, -10, -10, 4, -10\}$ può essere rappresentato, tra gli altri, dal seguente valore di tipo `int Multiset`:

```
Elemento(100,2,Elemento(2,1,Elemento(-10,3,Elemento(4,1,Vuoto)))).
```

Scrivere una funzione CAML

```
add:  'a multiset -> 'a -> 'a multiset
```

in modo che `(add m x)` sia il multinsieme ottenuto aggiungendo `x` al multinsieme `m`.
N.B.: in un valore di tipo `'a multiset` non possono esistere due occorrenze dello stesso valore di tipo `'a`. Ad esempio

```
Elemento(100,1,Elemento(100,1,Elemento(2,1,Elemento(-10,3,Elemento(4,1,Vuoto)))))
```

non è una corretta rappresentazione del precedente multinsieme in quanto l'elemento `100` è ripetuto due volte.

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

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

che, dato un array di interi e la sua dimensione, controlla che ogni valore dell'array, tranne l'ultimo, sia seguito nell'array da almeno un elemento strettamente minore. Si giochi d'astuzia!

ESERCIZIO 4 (punti 5)
Senza utilizzare ricorsione esplicita, definire in CAML una funzione che, data una lista di coppie di interi, restituisca la coppia con il secondo valore più grande. Se esistono nella lista più coppie con tale proprietà la funzione restituisce l'ultima tra di esse. La funzione non è definita sulla lista vuota.

ESERCIZIO 5 (punti 5)
Date le seguenti definizioni:

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

scrivere in `C` una procedura che, dati in ingresso attraverso opportuni parametri una lista di interi e un intero n, aggiunge n in fondo alla lista se n non è già presente in essa, lascia la lista inalterata altrimenti.