Fondamenti di Programmazione - CdL in MATEMATICA

Appello del 14/9/2015

num. eserc.	1	2	3	4
punt. tot	7	6	7	11

N.B.:

- Negli esercizi di programmazione, viene valutata anche la leggibilità del codice proposto.
- Inoltre, non è consentito l'uso di istruzioni che alterino il normale flusso dell'esecuzione (come, ad esempio, `continue`, `break` e istruzioni di `return` all'interno di cicli che ne provochino l'uscita forzata).
- Non è consentito l'uso di variabili statiche.
- Laddove è utilizzato, il tipo `boolean` è definito da `typedef enum {false,true} boolean`.

ESERCIZIO 1 (7 punti)
Dato il seguente automa non deterministico A sull'alfabeto $\Sigma = \{a, b, c, d\}$:

		a	b	c	d
I	q_0	$\{q_0, q_1\}$	$\{q_0\}$	$\emptyset$	$\emptyset$
	q_1	$\emptyset$	$\emptyset$	$\{q_2\}$	$\emptyset$
	q_2	$\emptyset$	$\emptyset$	$\emptyset$	$\{q_1, q_3\}$
F	q_3	$\{q_3\}$	$\emptyset$	$\emptyset$	$\emptyset$

- scrivere qual è il linguaggio accettato;
- disegnare l'automa deterministico equivalente risultante dall'applicazione dell'algoritmo di costruzione dei sottoinsiemi;
- fornire la **corrispondente grammatica regolare**.

ESERCIZIO 2 (6 punti)
Si scriva in C una funzione **iterativa** `boolean controlla(int v[], int dim)` che restituisce `true` se nell'array, che supponiamo sicuramente contenere il valore `0`, tutti gli elementi che seguono la prima occorrenza di `0` sono dispari.

ESERCIZIO 3 (7 punti)
Si scriva in C una funzione **ricorsiva** (che non deve pertanto fare uso di costrutti iterativi, nemmeno nelle sue sotto-funzioni, nel caso ce ne siano) che *legga da input* una sequenza di caratteri, che termina quando trova consecutivamente due occorrenze dello stesso carattere, e calcoli il numero di occorrenze di caratteri **non alfabetici** (l'ultimo carattere, uguale al precedente si può non contare anche se non alfabetico). La funzione, ad esempio, dovrebbe restituire `3` se la sequenza fosse:

```
'i'  K'  '9 '  ';'  's'  '?'  d'  'd'
```

ESERCIZIO 4 (11 punti)
Si supponga di lavorare con liste di interi così definite:

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

- Si scriva una procedura **ricorsiva** in C che, dati attraverso opportuni parametri una lista di interi e un intero $n > 0$, aggiunge un elemento, che contiene n, prima di ogni elemento pari presente nella lista.
- Scrivere una procedura **iterativa** in C che, dati attraverso opportuni parametri una lista di interi e un intero $n > 0$, elimina dalla lista l'ultimo elemento minore di n. Se nessun elemento della lista è minore di n, la procedura elimina, se esiste, l'ultimo elemento della lista.