Fondamenti di Programmazione - CdL in MATEMATICA

I Appello del 9/9/2013

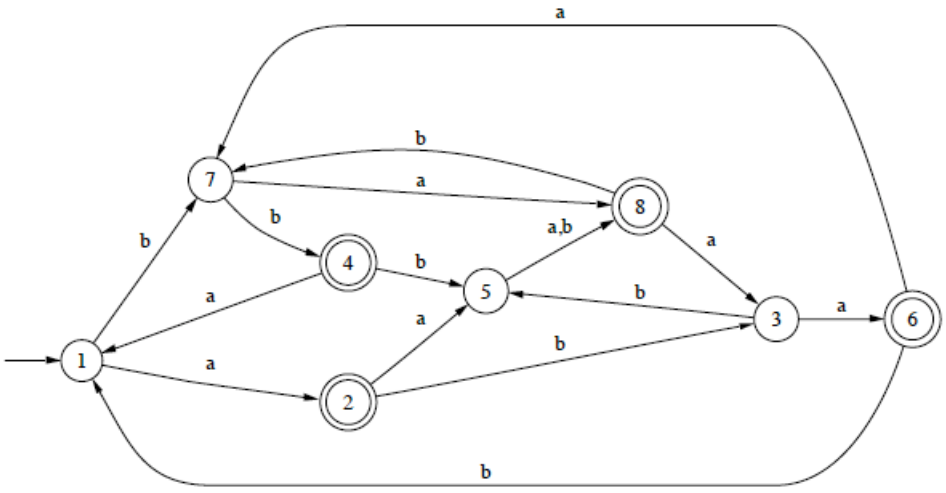
num. eserc.	1	2	3	4
punt. tot	7	7	7	9

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 a stati finiti deterministici.

- Si fornisca il corrispondente automa minimo M.
- Si derivi da M la grammatica regolare che generi il linguaggio riconosciuto da M.



ESERCIZIO 2 (7 punti)
Si consideri la seguente grammatica $G = \langle\{1,2,3\},\{S,A,B\},S,P\rangle$, in cui le produzioni in P sono:

$$\begin{array}{lcl} S & ::= & S3|A \\ A & ::= & 1A|B \\ B & ::= & 2B|2 \end{array}$$

Scrivere in C una funzione che dato un array di interi `v` e la sua dimensione `dim` restituisce `true` se la sequenza contenuta nell'array appartiene al linguaggio generato da G e restituisce `false` altrimenti.

ESERCIZIO 3 (7 punti)
Si scriva in C una funzione **ricorsiva** che legge in input una sequenza di interi terminati da zero e verifica se la somma degli elementi letti è pari, senza utilizzare la somma, ma ricordando l'aritmetica dei numeri pari e dispari, ovvero ricorrendo all'operazione di modulo e agli operatori logici per la composizione dei risultati parziali.

ESERCIZIO 4 (9 punti)
Si vuole modellare una lista di tessere relative ad una parola del gioco *Ruzzle*. Ogni tessera contiene (i) un campo per la lettera, (ii) un campo per il punteggio associato alla lettera (ii) uno che segnala per ogni posizione il bonus associato alla lettera (che può essere 2 a significare di raddoppiare e 3 a significare di triplicare, e 1 per segnalare che non c'è bonus) e (iv) un altro campo che rappresenta il bonus associato alla parola che passa da quella posizione (che può essere 2 a significare di raddoppiare e 3 a significare di triplicare, e 1 per segnalare che non c'è bonus). Nello schema della figura, ad esempio, la parola "misto" vale $(3+1+2+2+1)*2$, mentre la parola "rato" vale $((2+1+2+1)*3*2)$.

- Definire i tipi opportuni per rappresentare liste di questo tipo.
- Scrivere in C una funzione **iterativa** che, dati in ingresso una lista di tessere che rappresenta una parola valida ne calcoli il punteggio complessivo.
- Scrivere una procedura **ricorsiva** che data una lista di tessere che rappresenta una parola valida e due lettere, elimini dalla lista, se esistono, due tessere contigue che rappresentano le due lettere. Nello schema della figura si potrebbe per esempio chiedere di eliminare dalla parola "misto" le due lettere 's' e 't'. Attenzione, se troviamo la prima lettera, ma non la seconda subito dopo, la prima non va eliminata.