Fondamenti di Programmazione - CdL in MATEMATICA

II Prova di verifica del 30/5/2014

num. eserc.	1	2	3	4
punt. tot	8	6	7	9

N.B.: Negli esercizi di programmazione, vengono valutati anche l'uso delle condizioni booleane e la leggibilità del codice proposto. Inoltre, non è consentito l'uso di istruzioni che alterino il normale flusso dell'esecuzione all'interno di cicli e provochino l'uscita forzata. Laddove è utilizzato, il tipo `boolean` è definito da `typedef enum {false, true} boolean`.

ESERCIZIO 1 (8 punti)

- Scrivere una grammatica libera G_0 che generi il linguaggio $L_0 = \{w | w \in \{0,1\}^*$ e w contiene almeno due 1 consecutivi$\}$
- Scrivere una grammatica libera G_1 che generi il seguente linguaggio $L_1 = \{w | w \in \{0,1\}^*$ e w contiene più 1 che 0$\}$
- Un'espressione booleana `and-or` *ben formata* è definita dalle seguente regole:

  1. sia 0 che 1 sono espressioni ben formate;
  2. se A e B sono espressioni ben formate, lo sono anche $A \wedge B$ e $A \vee B$;
  3. se A è una espressione ben formata, lo è anche (A).

  La grammatica libera G_2 genera tutte le espressioni booleane `and-or` ben-formate con gli operatori binari $\wedge$ (and) e $\vee$ (or). L'insieme dei terminali è $\{0, 1, \wedge, \vee, (,)\}$ Le produzioni sono:

  $$B \rightarrow I | B \wedge B | B \vee B | (B)$$
  $$I \rightarrow 0 | 1$$

  1. Dimostrare che la grammatica G_2 è *ambigua*.
  2. Fornire una grammatica equivalente a G_2 che non sia ambigua.

ESERCIZIO 2 (6 punti)
Scrivere in C una funzione *iterativa* (non occorre scrivere il `main`) che legga da input una sequenza di interi positivi non decrescente che termina appena entra 0 e che restituisca il massimo numero di ripetizioni di uno stesso intero. Ad esempio, se la sequenza fosse `1 1 1 2 2 2 2 4 4 4 6 7 7 7 7 7 7 9 9 9 11 11 11 11 0`, la funzione dovrebbe restituire `6` che è il numero massimo di ripetizioni, relativo all'intero `7`.

ESERCIZIO 3 (7 punti)

- Scrivere in C una funzione *ricorsiva* `SommaCifre(n)` che prende in input un intero non negativo e restituisce la somma delle cifre che lo compongono. Per esempio, facendo l'invocazione `SommaCifre(1452)`, il risultato sarebbe: 1+4+5+7=12.
- La *radice digitale* o *numerica* di un numero intero `n` definita come il risultato della somma delle cifre che compongono il numero, reiterata fino ad ottenere un valore ad una sola cifra. Ad esempio, per calcolare la radice digitale di `1452` si procede in due passi consecutivi:

  1. 1+4+5+2=12
  2. 1+2 = 3

  – Scrivere in C una funzione *iterativa* `RadiceDigitaleIt(n)` che restituisce la radice digitale del numero intero `n` passato come parametro.
  – Scrivere in C una funzione puramente *ricorsiva* `RadiceDigitaleRic(n)` senza l'uso esplicito di costrutti iterativi.

(SEGUE ALTRA PAGINA →)

ESERCIZIO 4 (9 punti)
Si vuole modellare una lista di studenti, ordinata (in ordine crescente) rispetto al numero di matricola. Ogni elemento della lista deve contenere le seguenti informazioni:

- un intero che identifica il numero di matricola;
- un array di caratteri che identifica il cognome.

Per risolvere l'esercizio si facciano le seguenti cose.

- Definire i tipi opportuni per rappresentare una siffatta lista.
- Date due liste ordinate (senza ripetizioni) del tipo definito, si scriva una procedura (non occorre scrivere il `main`) per eliminare dalla prima lista tutti gli elementi che compaiono anche nella seconda lista.
- Data una lista del tipo definito e un numero intero `n`, si scriva una funzione (non occorre scrivere il `main`) che restituisca il numero di elementi le cui matricole siano maggiori o uguali ad `n`. N.B: non è detto che esistano numeri di matricole maggiori di `n`.