Esercitazione di laboratorio n. 5

(Caricamento sul portale entro le 23.59 del 21/11/2016 dell'esercizio 2 e di almeno uno tra gli esercizi 1 e 3)

Esercizio n. 1: Sequenza Q di Hofstadter

La sequenza Q di Hofstadter è una sequenza ricorsiva definita come:

$$Q(1) = Q(2) = 1$$
$$Q(n) = Q\big(n - Q(n-1)\big) + Q\big(n - Q(n-2)\big) \qquad \text{per n} > 2$$

I primi termini della sequenza sono:

```
1, 1, 2, 3, 3, 4, 5, 5, 6, 6, 6, 8, 8, 8, 10, 9, 10, 11, 11, 12, 12, 12, 12,
16, 14, 14, 16, 16, 16, 16, 20, 17, 17, 20, 21, 19, 20, 22, 21, 22, 23, 23,
24, 24, 24, 24, 24, 32, 24, 25, 30, 28, 26, 30, 30, 28, 32, 30, 32, 32, 32,
32, 40, 33, 31, 38, 35, 33, 39, 40, 37, 38, 40, 39, …
```

Si scriva un programma in C che, ricevuto un intero positivo `N`, stampi i primi N termini della sequenza stessa.

Esercizio n. 2: Conversione di base

Si scriva una funzione ricorsiva in C che, ricevuti tra i suoi parametri di ingresso 2 interi positivi e non nulli n e $b \geq 2$, calcoli e visualizzi la rappresentazione di n in base b sul minimo numero di cifre. Non è lecito utilizzare un vettore per accumulare le cifre calcolate ai vari passi.

Esercizio n. 3: Catalogo di prodotti (rivisto)

A partire dalle specifiche dell'esercizio n.3 del laboratorio 4, estendere le funzionalità del programma:

- per mantenere in contemporanea più ordinamenti per la struttura dati:
  - per prezzo, ascendente o discendente
  - per codice prodotto
  - per nome prodotto
- per inserire un nuovo prodotto
- per cancellare un prodotto esistente.

Osservazioni:

- per l'ordinamento: il vettore originale è letto una sola volta ed è mantenuto nell'esatto ordine di lettura per tutta la durata dell'esecuzione. Ad esso sono affiancati tanti vettori di puntatori a struttura quanti sono gli ordinamenti richiesti, con i quali sono gestiti gli ordinamenti stessi
- per l'inserzione: avviene al fondo del vettore originale con eventuale riallocazione nel caso il vettore sia pieno. Si ricordi di gestire opportunamente i vettori paralleli ordinati di puntatori
- per la cancellazione: la cancellazione è di tipo logico. A tale scopo, si introduca una variabile *flag* all'interno della `struct` originale, che marchi lo stato (cancellato/disponibile) per ogni prodotto della base dati.