Esercitazione di Laboratorio del 15/12/2016

Esercizio 1. Scrivere in C++ un programma che:

- legga da input una sequenza di numeri interi terminata da un numero negativo;
- memorizzi in un array tutti i numeri della sequenza che non sono multipli di 5;
- scandisca l'array e stampi su standard output tutte le sottosequenze terminate da 2, nonché tutte le posizioni in cui compare un "2".

Esempio: se la sequenza in input fosse la seguente:

3 5 6 8 25 8 2 4 30 6 2 5 6 10 2 7 9 -1,

allora nell'array si dovrebbe memorizzare la sequenza: 3 6 8 8 2 4 6 2 6 2 7 9; poi si dovrebbero stampare su standard output le 4 sottosequenze separate dai "2": {3 6 8 8}, {4 6}, {6} e {7 9}, e infine le posizioni occupate dai "2": 5, 8 e 10.

Esercizio 2. Si scriva un programma in C++ opportunamente modularizzato in funzioni che legga da input due sequenze (indicate di seguito come sequenza A e sequenza B) di numeri interi positivi, ciascuna terminata da uno zero (0), e stampi su standard output la stringa "OK" se esse godono della proprietà detta "star-max" (descritta di seguito), e stampi invece "NO" se la proprietà non è verificata.
Le due sequenze godono della proprietà "star-max" se tutti i numeri "star" della sequenza A sono più grandi di ogni numero della sequenza B. Un numero nella sequenza A è definito come "star" se esiste nella sequenza B almeno un numero che abbia il suo stesso numero di cifre.

Esempio: date le sequenze:

A = 1 1998 431 5 7361 701 0 B = 43 56 1976 67 0

i numeri "star" nella sequenza A sono 1998 e 7361; infatti sono composti di 4 cifre, e nella sequenza B c'è il numero 1976, anch'esso di 4 cifre. Nessun altro numero della sequenza A è "star": infatti sono tutti di una oppure tre cifre, e nessun numero nella sequenza B è composto di una o tre cifre (in effetti sono tutti di due o di quattro cifre). La proprietà "star-max" è in questo caso rispettata: infatti sia 1998 che 7361, i numeri "star" nella sequenza A, sono più grandi di tutti i numeri presenti nella sequenza B.

NOTA 1: Si può assumere che le sequenze abbiano una lunghezza massima pari a 100.
NOTA 2: Potrebbe essere utile prevedere una funzione che riceva un numero intero positivo e restituisca il numero delle cifre di cui esso è composto. Il suo prototipo potrebbe essere (ad esempio): `unsigned nCifre( int n );`
la funzione dovrebbe essere in grado di restituire il numero di cifre di "n", qualunque sia "n". Se si dovessero incontrare difficoltà nella stesura di tale funzione, è possibile assumere che i numeri nelle sequenze siano tutti minori di 10000 (quindi compresi tra 1 e 9999), e pertanto il numero di cifre possibili è compreso tra 1 e 4.

Esercizio 3. Si scriva un programma in C++ opportunamente modularizzato in funzioni che, letta da input una sequenza di numeri interi (che possono essere positivi o negativi) terminata dallo 0 (zero), individui i punti di tipo A e B descritti di seguito.
Un numero costituisce un punto di tipo A se è dispari, e il numero che lo precede e quello che lo segue sono pari; costituisce, invece, un punto di tipo B se è pari, e il numero che lo precede e quello che lo segue sono dispari.
Una volta individuati i punti di tipo A e B, il programma deve verificare se ogni punto di tipo A ha almeno un multiplo tra i punti di tipo B, e stampare su standard output la stringa “OK” in caso positivo; in caso negativo, si deve stampare “NO”.

Esempio: data la sequenza:

1 -4 11 3 70 5 -12 6 0,

l'unico punto di tipo A è il numero 5 (infatti è dispari, ed è preceduto da 70 e seguito da -12, che sono entrambi pari); i punti di tipo B sono invece -4 (preceduto da 1 e seguito da 5) e 70 (preceduto da 3 e seguito da 5). L'unico punto di tipo A, che è 5, ha come multiplo 70 che sta tra i punti di tipo B: pertanto il programma, in questo caso, dovrebbe stampare “OK”.

Esercizio 4. Scrivere un programma che implementi la “mescolata perfetta” di un mazzo di carte con un numero pari di carte. Da notare che la mescolata perfetta consiste nel lasciare invariati i due estremi del mazzo e poi mettere la prima metà delle carte nelle posizioni pari del mazzo mescolato e la seconda metà delle carte nelle posizioni dispari del mazzo mescolato.

Esempio: Il programma dovrà sostituire {11,22,33,44,55,66,77,88} con {11,55,22,66,33,77,44,88}. Il programma, ovviamente, dovrà leggere da input una sequenza pari di numeri interi e memorizzarla in un array di dimensione prefissata.

NOTA: si modularizzi opportunamente il programma in funzioni, prevedendo un metodo per il mescolamento dell'array. Questa funzione dovrà interfogliare la prima metà di un array con la seconda metà: ciò si ottiene facilmente usando un array temporaneo temp*, e copiando alla fine* temp *nell'array originario* a*.*