La sequenza di esercizi che segue è in ordine crescente di difficoltà. Un buon obbiettivo è quello di arrivare a terminare i primi 2 e almeno comprendere il meccanismo dietro all'esercizio 2b. L'esercizio 3 è di livello medio-basso.

Es. 1: Inversione array

Sia A un vettore di n interi dato in input, le cui celle sono $A[0]A[1]\cdots A[n-1]$. Il vettore invertito A^R è composto dalle celle di A lette da destra verso sinistra: $A[n-1]A[n-2]\cdots A[1]A[0]$. Il vostro programma deve

- Permettere l'inserimento di A da tastiera. Per fare questo, deve inizialmente leggere il valore n da tastiera. Poi deve leggere gli n interi. Questa prima fase deve avvenire in una funzione a parte.
- Invertire l'array A *in loco*, ovvero senza usare altri array di supporto.
- Stampare l'array A^R, utilizzando una funzione a parte, visualizzando su una sola riga tutti i valori.

Potete sempre assumere che $1 \leq n \leq 10000$.

Esempio

6
4 1 3 0 0 2
2 0 0 3 1 4

5
1 2 3 4 5
5 4 3 2 1

Es. 2: Permutazione casuale

Sia n un intero. Il *mescolamento alla Fisher-Yates* è un metodo per generare una permutazione della sequenza $0, 1, 2, \ldots, n-1$ molto efficace e non distorto (nel senso che tutte le permutazione hanno uguale probabilità di essere generate). L'algoritmo funziona nel seguente modo. Dato n si inizializza un array A con i valori da 0 a $n-1$ in ordine. Poi si esegue la seguente procedura per ogni posizione i, con $1 \leq i \leq n$, in ordine:

- Si sceglie un valore j a caso tra i e $n-1$ compresi.

- Si scambiano i contenuti delle celle i e j di A.

Per tanto, il programma parte dalla prima posizione e sceglie con quale scambiarla tra le successive (se stessa inclusa), poi si sposta alla seconda cella e così via fino al termine.
Scrivete un programma che permetta di inserire n e generi una permutazione casuale. Assicuratevi che il vostro programma, tra varie esecuzioni, non generi sempre la stessa permutazione.
Potete sempre assumere che $1 \leq n \leq 10000$.

Esempio

Nota: trattandosi di un programma che utilizza numeri pseudocasuali, qualsiasi output conforme alle specifiche è possibile.

14
1 10 11 5 7 12 8 9 2 0 3 4 6 13

8
7 5 1 3 2 6 4 0

Es. 2bis

L'esercizio 1 e l'esercizio 2 sono una buona coppia per il test automatico dei vostri esercizi, pratica molto utili alla ricerca di errori. Lo scopo è usare l'esercizio 2 come generatore di sequenze di input di grandi dimensioni per testare l'inversione degli array. Combinate il codice scritto fin'ora nella seguente maniera:

- Leggete un intero n da tastiera.
- Generate una permutazione casuale di lunghezza n senza stamparla a video, in un array di interi chiamato A.
- Copiate il contenuto di A in un altro array B.
- Eseguite *due volte* il codice di inversione su B. In questo modo, se l'inversione è corretta, il risultato sarà esattamente ancora A.
- Controllate, cella per cella, che l'array B invertito due volte corrisponda all'array A e stampate a video il verdetto.

Es. 3: L'elemento mancante

Sia n un intero. Sia S la sequenza $0, 1, \ldots, n-1$. L'utente elimina uno ed uno solo degli n elementi di S e poi inserisce gli elementi rimanenti in un qualche ordine a suo piacimento.
Scrivete un programma che determini l'elemento mancante. Progettate diverse soluzioni, partendo dalla più semplice, cercando via via di utilizzare sempre meno memoria per ottenere il risultato. In particolare:

- Progettate una soluzione che funzioni memorizzando la sequenza in input.
- Progettate una soluzione che funzioni senza memorizzare la sequenza in input.