Os menores correm mais rápido!

Como se sabe, o *bubble sort* clássico faz trocas de modo que os elementos maiores sempre chegam mais rápido à sua posição definitiva. Neste exercício, você deve construir uma variação do *bubble sort* que desloca, primeiro, os elementos menores (ao invés dos maiores). Para cada rastreamento completo do vetor, você deve imprimir sua configuração. Você deve interromper o algoritmo assim que o vetor ficar ordenado (ou seja, não deve haver impressões repetidas do vetor já ordenado).

Entrada

A entrada consiste em 2 linhas. A primeira linha possui um número inteiro N (0 < N < 1024), tal que N é a quantidade de elementos do vetor. A segunda linha possui uma sequência de N números inteiros, que representam os números que devem ser armazenados no vetor. O armazenamento deve seguir a ordem em que os elementos são apresentados.

Saída

A saída possui M linhas (0 < M < (N+1)). Cada linha deve mostrar a sequência de elementos no vetor após uma varredura de trocas do algoritmo bubble sort. A primeira linha possui a configuração inicial do vetor. A última linha possui a configuração do vetor já ordenado. Não deve haver linhas iguais.

Exemplo:

Entrada	Saída
10 4 -3 15 10 12 8 7 9 5 1	4 -3 15 10 12 8 7 9 5 1 -3 4 1 15 10 12 8 7 9 5 -3 1 4 5 15 10 12 8 7 9 -3 1 4 5 7 15 10 12 8 9 -3 1 4 5 7 8 15 10 12 9 -3 1 4 5 7 8 9 15 10 12 -3 1 4 5 7 8 9 10 15 12 -3 1 4 5 7 8 9 10 12 15

Crescente ou Decrescente? Talvez nenhum deles!

Faça um programa que receba vários vetores de números inteiros e determina se eles estão ordenados ou não. A ordenação pode ser crescente (menores antes dos maiores) ou decrescente (maiores antes dos menores).

Entrada

O programa terá vários casos de teste. Cada caso de teste é composto por exatamente 2 linhas. A primeira linha do caso de teste contem um inteiro N, $0 \leq N \leq 10000$, representando o tamanho do vetor. A segunda linha do caso de teste contém N números inteiros separados por um espaço em branco cada. A entrada termina com EOF.

Saída

O programa gera várias linhas de saída, uma para cada caso de teste informado na entrada. Caso o vetor esteja ordenado de forma crescente imprima uma letra C. Caso o vetor esteja ordenado de forma decrescente, imprima D. Caso o vetor não esteja ordenado, imprima a letra N. Após a impressão de cada letra, inclusive da última, salte uma linha.

Exemplo:

Entrada	Saída
10 0 1 2 3 4 5 6 7 8 9 6 9 5 0 2 4 1 8 1 2 3 4 5 6 7 0 10 9 8 7 6 5 4 3 2 1 0 7 62 74 96 103 540 599 853	C N N D C

Selection Sort

Faça um programa que receba um vetor de inteiros de tamanho N e ordene os seus elementos usando o algoritmo de seleção (Selection Sort).

Entrada

O programa terá apenas um caso de teste. A primeira linha do caso de teste contem um inteiro N, 0 < N ≤ 10000, representando o tamanho do vetor. A segunda linha do caso de teste contem N números inteiros separados por um espaço em branco cada.

Saída

O programa gerá várias linhas de saída. A primeira linha, trata-se do vetor na sua ordem original, com os valores separados por espaço embranco. A cada troca que o algoritmo realizar, você deve imprimir todo o vetor, com seus valores separados por espaço em branco. A penúltima linha trata-se do vetor após passar pelo algoritmo de ordenação, ou seja, ordenado com seus valores separados por espaço em branco. A última linha trata-se de um inteiro representando o número de trocas (swap) entre elementos no vetor. Vale lembrar que antes do primeiro elemento do vetor não há nada impresso, assim como não há nada impresso depois do último elemento do vetor apenas uma quebra de linha. Após a impressão da última linha da saída, salte uma linha.

Exemplo:

Entrada	Saída
10 54 12 2 90 4 78 55 21 2 48	54 12 2 90 4 78 55 21 2 48 2 12 54 90 4 78 55 21 2 48 2 2 54 90 4 78 55 21 12 48 2 2 4 90 54 78 55 21 12 48

	2 2 4 12 54 78 55 21 90 48 2 2 4 12 21 78 55 54 90 48 2 2 4 12 21 48 55 54 90 78 2 2 4 12 21 48 54 55 90 78 2 2 4 12 21 48 54 55 90 78 2 2 4 12 21 48 54 55 78 90 2 2 4 12 21 48 54 55 78 90 8

Insertion Sort

Faça um programa que receba um vetor de inteiros de tamanho N e ordene os seus elementos usando o algoritmo de inserção (Insertion Sort).
Entrada
O programa terá apenas um caso de teste. A primeira linha do caso de teste contem um inteiro N, 0 < N ≤ 10000, representando o tamanho do vetor. A segunda linha do caso de teste contem N números inteiros separados por um espaço em branco cada. É garantido que não há repetição dos valores do vetor.
Saída
O programa gerará várias linhas de saída. A primeira linha, trata-se do vetor na sua ordem original, com os valores separados por espaço em branco. A cada troca que o algoritmo realizar, você deve imprimir todo o vetor, com seus valores separados por espaço em branco. A penúltima linha trata-se do vetor após passar pelo algoritmo de ordenação, ou seja, ordenado com seus valores separados por espaço em branco. A última linha trata-se de palavra “Trocas:X” seguida da quantidade de trocas que o algoritmo realizou. Observe que a letra T está em maiúsculo e após os dois pontos não há um espaço em branco antes do número inteiro X.
Após a impressão da última linha da saída, salte uma linha.

Exemplo:

Entrada	Saída
10 54 12 2 90 4 78 55 21 1 48	54 12 2 90 4 78 55 21 1 48 12 54 2 90 4 78 55 21 1 48 12 2 54 90 4 78 55 21 1 48 2 12 54 90 4 78 55 21 1 48 2 12 54 4 90 78 55 21 1 48 2 12 4 54 90 78 55 21 1 48 2 4 12 54 90 78 55 21 1 48 2 4 12 54 78 90 55 21 1 48 2 4 12 54 78 55 90 21 1 48 2 4 12 54 55 78 90 21 1 48 2 4 12 54 55 78 21 90 1 48 2 4 12 54 55 21 78 90 1 48 2 4 12 54 21 55 78 90 1 48 2 4 12 21 54 55 78 90 1 48

	2 4 12 21 54 55 78 1 90 48 2 4 12 21 54 55 1 78 90 48 2 4 12 21 54 1 55 78 90 48 2 4 12 21 1 54 55 78 90 48 2 4 12 1 21 54 55 78 90 48 2 4 1 12 21 54 55 78 90 48 2 1 4 12 21 54 55 78 90 48 1 2 4 12 21 54 55 78 90 48 1 2 4 12 21 54 55 78 48 90 1 2 4 12 21 54 55 48 78 90 1 2 4 12 21 54 48 55 78 90 1 2 4 12 21 48 54 55 78 90 1 2 4 12 21 48 54 55 78 90 Trocas:25

Quicksort

Implemente o algoritmo de ordenação Quick Sort (versão que seleciona o último elemento do vetor como pivô) e imprima o número total de comparações necessárias para ordenar um vetor de números inteiros.

Obs. Não use variáveis globais!

Entrada:

O programa terá apenas um caso de teste.A primeira linha do caso de teste é um número inteiro N, 1 ≤ N ≤ 10000. Na linha seguinte serão dados N inteiros separados por um espaço em branco cada.

Saída:

O programa gera como saída três linhas.

A primeira linha trata-se do vetor original antes de invocar o procedimento para ordenação.

A segunda linha trata-se da impressão do vetor após o termino do procedimento de ordenação. Quebre uma linha ao final.

A terceira linha trata-se do número de comparações necessárias para ordenar o vetor. Apenas conte o número de comparações em que pelo menos um elemento do vetor seja utilizado na comparação.

Exemplo:

Entrada	Saída
10 9 65 12 7 21 5 7 46 52 10	9 65 12 7 21 5 7 46 52 10 5 7 7 9 10 12 21 46 52 65 Comparacoes: 20
1 20	20 20 Comparacoes: 0
10 0 1 2 3 4 5 6 7 8 9	0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9

	Comparacoes: 45

Ordenação Parcial

Modifique o algoritmo de ordenação por seleção (Selection Sort) para obter uma sequência ordenada parcial contendo apenas os k menores elementos da sequência ordenados na sua forma crescente.

Entrada

O programa terá apenas um caso de teste.A primeira linha do caso de teste é um número inteiro N, 1 ≤ N ≤ 10000. Na linha seguinte serão dados N inteiros separados por um espaço em branco cada. A terceira linha um número k (1 ≤ k ≤ N).

Saída

O programa gera como saída duas linhas. A primeira linha trata-se do vetor original antes de invocar o procedimento para ordenação parcial. A segunda linha trata-se da impressão do vetor após o termino do procedimento de ordenação. Salte uma linha ao final.

Exemplo:

Entrada	Saída
10 9 65 12 7 21 5 7 46 52 10 3	9 65 12 7 21 5 7 46 52 10 5 7 7 65 21 9 12 46 52 10
10 9 65 12 7 21 5 7 46 52 10 5	9 65 12 7 21 5 7 46 52 10 5 7 7 9 10 65 12 46 52 21
10 9 65 12 7 21 5 7 46 52 10 9	9 65 12 7 21 5 7 46 52 10 5 7 7 9 10 12 21 46 52 65