Clase 3 - OCILabs

Ejercicio 1: Utilizar y modificar los algoritmos

Su hermano chico tomó un curso de programación en el colegio y tiene una tarea que le entregó su profesora. La tarea consiste en ordenar diferentes secuencias. Tú decides ayudarlo, emocionado o con tus conocimiento aprendidos en el taller OCILabs.

Primera pregunta

Elija uno de los siguiente tres algoritmos: Bubble Sort, Insertion Sort y Selection Sort. Realice un programa que haga uso del algoritmo elegido para ordenar una secuencia de números.

Para esto considere el siguiente input:

En la primera línea se encuentra un entero n, correspondiente a la cantidad de elementos en la secuencia. Luego, en la siguiente línea se encuentra n enteros separados por un espacio.

Input de ejemplo

```
10
3 2 10 -2 5 8 7 1 2 -3
```

Su programa debe imprimir la secuencia de números, pero ordenada de menor a mayor.

Output de ejemplo

```
-3 -2 1 2 2 3 5 8 7 10
```

Segunda pregunta

Al programa anterior realizado, debe modificarlo de tal forma que ahora los elementos se ordenen de mayor a menor y no de mayor a menos.

Para esta pregunta tome en cuenta el mismo input que la pregunta anterior.

Input de ejemplo

```
10
3 2 10 -2 5 8 7 1 2 -3
```

Output de ejemplo

```
-3 -2 1 2 2 3 5 8 7 10
```

Tercera pregunta

Ordenar no es algo que solo se puede aplicar a números, si no que también es algo que se puede aplicar a diferentes tipos de datos y estructuras. Por ejemplo, se pueden ordenar caracteres, palabras, vectores,

etcétera.

En esta ocasión se le pide modificar el programa realizado en la pregunta de tal manera que se puedan ordenar caracteres en vez de números.
Para esta pregunta considere el siguiente input:

En la primera línea se encuentra un entero n, correspondiente a la cantidad de elementos en la secuencia. Luego, en la siguiente línea se encuentra n caracteres separados por un espacio.

Input de ejemplo

```
10
d f h n m c e l a i
```

Su programa debe imprimir la secuencia de caracteres ordenada alfabéticamente.

Output de ejemplo

```
a c d e f h i l m n
```

NOTA: recordar que los caracteres siguen la notación ASCII, notación estudiada en la primera clase del taller avanzado.

Cuarta pregunta

Siguiendo la idea de la pregunta anterior, se le pide ordenar strings. Para ordenar strings no se puede hacer uso de las operaciones para comparar como: <=, <, ==, > y >=.

Pero entonces, ¿Cómo se puede ordenar palabras? Para esto se seguirá el siguiente algoritmo, utilicemos como ejemplo los siguientes dos strings *abc* y *abaa*

- Se toma el primer carácter de cada palabra. Para *abc* es *a* y *abaa* es *a*. Al ser ambos caracteres iguales, se debe revisar el segundo caracter.
- Se toma el segundo carácter de cada palabra. Para *abc* es *b* y *abaa* es *b*. Al ser ambos caracteres iguales, se debe revisar el tercer carácter.
- Se toma el tercer carácter de cada palabra. Para *abc* es *c* y *abaa* es *a*. En este caso, ya no son iguales por lo que no se debe revisar más caracteres. El tercer carácter de *abaa* es menor alfabéticamente a *c*, entonces se puede decir que *abaa* es *"menor"* a *abc*.

Otro ejemplo sería *aba* y *abaa*:

- Se toma el primer carácter de cada palabra. Para *aba* es *a* y *abaa* es *a*. Al ser ambos caracteres iguales, se debe revisar el segundo caracter.
- Se toma el segundo carácter de cada palabra. Para *aba* es *b* y *abaa* es *b*. Al ser ambos caracteres iguales, se debe revisar el tercer carácter.
- Se toma el tercer carácter de cada palabra. Para *aba* es *a* y *abaa* es *a*. Ambos caracteres son iguales, PERO *aba* es más corto, entonces se considera *aba* es *"menor"* a *abaa*.

Para esta pregunta se considerara el siguente input:

En la primera línea se encuentra un entero n, correspondiente a la cantidad de elementos en la secuencia. Luego, en la siguiente línea se encuentra n strings separados por un espacio.

Input de ejemplo

```
7
magic ocilabs mario otravez marie ocilab gamer
```

Su programa debe imprimir la secuencia de strings ordenada según lo explicado previamente.

Output de ejemplo

```
gamer magic marie mario ocilab ocilabs otravez
```

A - Mjehuric

Goran tiene cinco piezas de madera dispuestas en secuencia. Hay un número entre 1 y 5 inscrito en cada pieza, de modo que cada número aparece exactamente en una de las cinco piezas. Goran quiere ordenar las piezas para formar la secuencia 1,2,3,4,5 y lo hace así:

- Si el número de la primera pieza es mayor que el número de la segunda pieza, cámbielos.
- Si el número de la segunda pieza es mayor que el número de la tercera pieza, cámbielos.
- Si el número de la tercera pieza es mayor que el número de la cuarta pieza, cámbielos.
- Si el número de la cuarta pieza es mayor que el número de la quinta, cámbielos.
- Si las piezas no forman la secuencia 1,2,3,4,5, vaya al paso 1.

Input

La primera línea contiene cinco números enteros separados por espacios simples, el orden de las piezas. Los números estarán entre 1 y 5 (inclusive) y no habrá duplicados. El pedido inicial no será 1,2,3,4,5.

Output

Después de intercambiar dos piezas, muestre por pantalla el pedido de las piezas en una sola línea separada por espacios.

Ejemplo

Input

```
2 1 5 3 4
```

Output

```
1 2 5 3 4
1 2 3 5 4
1 2 3 4 5
```

Input

```
2 3 4 5 1
```

Output

```
2 3 4 1 5
2 3 1 4 5
2 1 3 4 5
1 2 3 4 5
```

B - Helpful Maths

Xenia, la matemática principiante, es una estudiante de tercer año en la escuela primaria. Ahora está aprendiendo la operación de suma.

El maestro ha escrito la suma de varios números. Los alumnos deben calcular la suma. Para facilitar el cálculo, la suma solo contiene los números 1, 2 y 3. Aún así, eso no es suficiente para Xenia. Ella solo está comenzando a contar, por lo que puede calcular una suma solo si los sumandos siguen en orden no decreciente. Por ejemplo, no puede calcular la suma $1+3+2+1$ pero puede calcular las sumas $1+1+2$ y $3+3$.

Tienes la suma que estaba escrita en la pizarra. Reorganice las sumas e imprima la suma de tal manera que Xenia pueda calcular la suma.

Input

La primera línea contiene una cadena s no vacía - la suma que Xenia necesita contar. La cadena s no contiene espacios. Solo contiene dígitos y caracteres "+". Además, la cadena s es una suma correcta de los números 1, 2 y 3. La cadena s tiene una longitud máxima de 100 caracteres.

Output

Muestre por pantalla la nueva suma que Xenia pueda contar.

Ejemplo

Input

```
3+2+1
```

Output

```
1+2+3
```

Input

```
1+1+3+1+3
```

Output

```
1+1+1+3+3
```

Input

```
2
```

Output

```
2
```

C - Train Swapping

En una antigua estación de tren, es posible que todavía se encuentre con uno de los últimos "intercambiadores de trenes" que quedan. Un intercambiador de trenes es un empleado del ferrocarril, cuyo único trabajo es reorganizar los vagones de los trenes.

Una vez que los vagones están dispuestos en el orden óptimo, todo lo que tiene que hacer el conductor del tren es dejar los vagones, uno por uno, en las estaciones para las que está destinada la carga.

El título "intercambiador de trenes" proviene de la primera persona que realizó esta tarea, en una estación cercana a un puente ferroviario. En lugar de abrirse verticalmente, el puente giraba alrededor de un pilar en el centro del río. Después de girar el puente 90 grados, los barcos podrían pasar a la izquierda o a la derecha.

El primer intercambiador de trenes había descubierto que el puente podía funcionar con dos vagones como máximo. Al girar el puente 180 grados, los vagones cambiaron de lugar, lo que le permitió reacomodar los vagones (como efecto secundario, los vagones miraron en la dirección opuesta, pero los vagones de tren pueden moverse en ambos sentidos, así que a quién le importa).

Ahora que casi todos los intercambiadores de trenes han desaparecido, a la compañía ferroviaria le gustaría automatizar su operación. Parte del programa a desarrollar, es una rutina que decide para un tren dado el menor número de permutas de dos vagones adyacentes necesarios para ordenar el tren. Tu tarea es crear esa rutina.

Input

La entrada contiene en la primera línea el número de casos de prueba (N). Cada caso de prueba consta de dos líneas de entrada. La primera línea de un caso de prueba contiene un entero L, que determina la longitud del tren ($0 \leq L \leq 50$). La segunda línea de un caso de prueba contiene una permutación de los números del 1 al L, lo que indica el orden actual de los carros. Los carros deben pedirse de manera que el carro 1 venga primero, luego el 2, etc. y el carro L llegue al final.

Output

Para cada caso de prueba, muestre por pantalla la sentencia: "Optimal train swapping takes S swaps" donde S es un entero.

Ejemplo

Input

```
3
3
1 3 2
4
4 3 2 1
2
2 1
```

Output

```
Optimal train swapping takes 1 swaps.
Optimal train swapping takes 6 swaps.
Optimal train swapping takes 1 swaps.
```

D - Birthdates

Escriba un programa que identifique a la persona más joven y a la persona más mayor en una clase.

Input

El número n ($1 \leq n \leq 100$) en la primera línea determina el número de personas en la clase. Las siguientes n líneas contienen el nombre de la persona y su fecha de nacimiento, con el siguiente formato:

nombre dd mm aaaa

En donde "nombre" es una sola palabra de menos de 15 letras y "dd" corresponde al día del mes, "mm" al número del mes y "aaaa" al año de la fecha de nacimiento de la persona.

Suponga que no hay dos personas que tengan la misma fecha de nacimiento ni que tengan el mismo nombre.

Output

Muestre por pantalla 2 líneas que contengan el nombre de la persona más joven y el de la persona más mayor, respectivamente.

Ejemplo

Input

```
5
Mickey 1 10 1991
Alice 30 12 1990
Tom 15 8 1993
Jerry 18 9 1990
Garfield 20 9 1990
```

Output

```
Tom
Jerry
```