> A - Ordenamiento por selección

Tienes que programar el algoritmo de ordenamiento por selección y tienes que decir la cantidad de intercambios que se hacen al ordenar una secuencia de elementos.

* **Input:**

La primera línea de la entrada incluye un número entero N, $(1 \leq N \leq 100)$, el número de elementos en la secuencia.

En la segunda línea, se proporcionan N elementos de la secuencia separados por espacios.

* **Output:**

La salida consta de 2 líneas.

En la primera línea, imprime la secuencia ordenada. Dos elementos contiguos de la secuencia deben estar separados por un espacio.

En la segunda línea, imprime el número de operaciones de intercambio.

* **Ejemplo**

Input:

```
6
5 6 4 2 1 3
```

Output:

```
6
5 6 4 2 1 3
```

Input:

```
6
5 2 4 6 1 3
```

Output:

```
1 2 3 4 5 6
3
```

> B - Ordenamiento por inserción

El algoritmo de Insertion Sort es una técnica clásica de ordenamiento. Tienes que implementar el algoritmo de ordenamiento por inserción y decir cuantos intercambios se realizaron entre dos pares de numeros.

* Input:

La primera línea contiene el número de casos de prueba T, $(1 \leq T \leq 5)$. A continuación, se presentan T casos de prueba.

La primera línea de cada caso incluye un número entero N, $(1 \leq N \leq 10^5)$, el número de elementos en la secuencia.

En la segunda línea de cada caso, se proporcionan N elementos de la secuencia separados por espacios.

* Output:

Por cada caso de prueba, tienes que imprimir la cantidad de intercambios que se han realizado por secuencia.

* Ejemplo

Input:

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

Output:

```
0
4
```

> C - Joaquín y las piezas de madera

Joaquín tiene cinco piezas de madera dispuestas en una secuencia. Hay un número entre 1 e 5 grabado en cada pieza, de modo que cada número aparece exactamente en una de las cinco piezas. Joaquín desea ordenar las piezas para formar la secuencia $1, 2, 34, 5$ y lo hace de la siguiente manera:

- Si el número en la primera pieza es mayor que el número en la segunda pieza, los intercambia.
- Si el número en la segunda pieza es mayor que el número en la tercera pieza, los intercambia.
- Si el número en la tercera pieza es mayor que el número en la cuarta pieza, los intercambia.
- Si el número en la cuarta pieza es mayor que el número en la quinta pieza, los intercambia.
- Si las piezas no forman la secuencia $1, 2, 3, 4, 5$, regresa al paso 1.

Escribe un programa que, dada la disposición inicial de las piezas, muestre la disposición después de cada intercambio.

* Input:

La primera línea contiene cinco enteros separados por espacios individuales, la disposición de las piezas. Los números estarán entre 1 e 5 (inclusive) y no habrá duplicados. La disposición inicial no será $1, 2, 3, 4, 5$.

* Output:

Después de intercambiar dos piezas, muestra la disposición de las piezas en una sola línea, separadas 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
```

> D - Orden de altura

José siempre hace que su clase se forme en orden de altura (el más bajo al frente de la línea). Cada septiembre llega una nueva clase de exactamente 20 estudiantes de tercero básico, todos con alturas distintas.

Durante los primeros días, lleva mucho tiempo poner a los niños en orden de altura, ya que nadie sabe dónde deben estar en la línea. No hace falta decir que hay mucho movimiento y reajuste.

Este año, José decidió probar un nuevo método para minimizar este caos en el ordenamiento. Se seleccionaría a un estudiante para que sea la primera persona en la línea. Luego, se seleccionaría otro estudiante que buscaría a la primera persona en la línea que sea más alta que él, y se colocaría frente a esa persona, lo que haría que todos los estudiantes detrás de él retrocedieran para darle espacio. Si no hay ningún estudiante más alto, entonces él se colocaría al final de la línea.

Este proceso continúa, un estudiante a la vez, hasta que todos los estudiantes estén en línea, momento en el cual los estudiantes estarán ordenados por altura.

Para este problema, escribirás un programa que calcule el número total de pasos dados hacia atrás durante el proceso de ordenamiento para una clase dada de estudiantes.

* **Input:**

La primera línea de entrada contiene un solo número entero T $(1 \leq T \leq 1000)$, que es el número de casos de prueba que le siguen. Cada caso de prueba debe procesarse de forma independiente.

Cada caso de prueba consta de una sola línea de entrada. Contiene el número del caso de prueba (N), seguido de 20 enteros no negativos y únicos separados por un solo espacio. Los 20 enteros representan la altura (en milímetros) de cada estudiante en la clase.

* **Output:**

Para cada caso de prueba, hay una línea de salida. La línea de salida única consta del numero del caso de prueba, N, seguido de un espacio y luego del número total de pasos dados hacia atrás.

* **Ejemplo**

Input:

```
4
1 900 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919
2 919 918 917 916 915 914 913 912 911 910 909 908 907 906 905 904 903 902 901 900
3 901 902 903 904 905 906 907 908 909 910 911 912 913 914 915 916 917 918 919 900
4 918 917 916 915 914 913 912 911 910 909 908 907 906 905 904 903 902 901 900 919
```

Output:

```
1 0
2 190
3 19
4 171
```