Permutations
Problem ID: a10p06permutations

Let us say two numbers are permutations of each other if it is possible to rearrange the digits of one to get the other.

For example, 3511 and 3151 are permutations of each other, but 3151 and 6329 are not. Note that 3511 and 313 are not permutations even though the contain the same set of digits. Same goes for 3511 and 135.

Write a program that, given two numbers a and b, answers whether they are permutations of each other.

The program should be split into functions, with a function called `are_permutation_of_same_digits(m, n)`, which returns `True` or `False` based on whether m and n are permutations of each other.

Although the implementation of the function can be short, and can easily be incorporated directly back into the calling code, it can be useful to isolate a part of the problem like this while developing the solution, so that it does not muddle the overall thought process. Then you can focus on how to solve this isolated problem, without getting distracted by the overall context.

Hints

1. You can convert an integer to a string to get a string of digits (or just not convert the input to integers to begin with), and then you can convert the string of digits to a list to be able to rearrange the order of the digits.

2. To check if it is possible to rearrange the digits of one number to get another, there might be a lot of possible rearrangements, and not obvious how to check them all.

   Perhaps you can find a simpler way, by using some kind of canonical representative for the entire set of numbers that are permutations of a particular number. Then, to check if two numbers are permutations of each other, you can find the canonical representative of each number, and compare those instead. If both numbers have the same canonical representative, then they must both be permutations of that representative, and hence of each other. What rearrangement of the digits, independent of the original arrangement of the digits, can you use as such a canonical representative?

3. In case you are wondering what these type indicators are doing there, in the line

   ```
   def are_permutation_of_same_digits(m: str, n: str) -> bool:
   ```

   these are so called annotations (also called type hints), that specify what type of values the function expects to receive, and what the type of the return value should be. Python does not enforce this, which is to say it is possible to send values of different types to the function, but if you use these consistently, then IDEs like VSCode can aid you in keeping track of the type of each variable, and warn you when a wrong type is being passed into a function etc.

   In addition, the annotations inform the IDE about the type of a variable, allowing it to assist you with improved autocomplete functionality, like suggesting available methods on objects when you add a dot (.) after a variable etc.

Input

The input consists of two lines.
The first line contains an integer a, with $0 \leq a \leq 10^8$.
The first line contains an integer b, with $0 \leq b \leq 10^8$.

Output

The output consists of one line. `The numbers {a} and {b} are permutations of each other.` or `The numbers {a} and {b} are not permutations of each other.` depending on if a and b are permutations of each other.

Sample Input 1 **Sample Output 1**

3511 3151	The numbers 3511 and 3151 are permutations of each other.

Sample Input 2 **Sample Output 2**

3511 6329	The numbers 3511 and 6329 are not permutations of each other.

Sample Input 3 **Sample Output 3**

3511 135	The numbers 3511 and 135 are not permutations of each other.