Given a String input1, which contains many number of words separated by : and each word contains exactly two lower case alphabets, generate an output based upon the below 2 cases.

Note:

1. All the characters in input 1 are lowercase alphabets.
2. input 1 will always contain more than one word separated by :
3. Output should be returned in uppercase.

Case 1:

Check whether the two alphabets are same.

If yes, then take one alphabet from it and add it to the output.

Example 1:

input1 = ww:ii:pp:rr:oo

output = WIPRO

Explanation:

word1 is ww, both are same hence take w

word2 is ii, both are same hence take i

word3 is pp, both are same hence take p

word4 is rr, both are same hence take r

word5 is oo, both are same hence take o

Hence the output is WIPRO

Case 2:

If the two alphabets are not same, then find the position value of them and find maximum value – minimum value.

Take the alphabet which comes at this (maximum value - minimum value) position in the alphabet series.

Example 2"

input1 = zx:za:ee

output = BYE

Explanation

word1 is zx, both are not same alphabets

position value of z is 26

position value of x is 24

max – min will be 26 – 24 = 2

Alphabet which comes in 2nd position is b

Word2 is za, both are not same alphabets

position value of z is 26

position value of a is 1

max – min will be 26 – 1 = 25

Alphabet which comes in 25th position is y

word3 is ee, both are same hence take e

Hence the output is BYE

For example:

Input	Result
ww:ii:pp:rr:oo	WIPRO
zx:za:ee	BYE

```
import java.util.Scanner;
public class prog{
   public static void main(String[] args) {
      Scanner sc = new Scanner(System.in);
      String input = sc.nextLine();
      String[] words = input.split(":");
      StringBuilder result = new StringBuilder();
      for (String word : words) {
         char c1 = word.charAt(0);
         char c2 = word.charAt(1);
         if (c1 == c2) {
            result.append(Character.toUpperCase(c1));
         } else {
            int pos1 = c1 - 'a' + 1;
            int pos2 = c2 - 'a' + 1;
            int diff = Math.abs(pos1 - pos2);
            char newChar = (char) ('a' + diff - 1);
            result.append(Character.toUpperCase(newChar));
         }
      }
      System.out.println(result.toString())
   }
}
```

	Input	Expected	Got	
✓	ww:ii:pp:rr:oo	WIPRO	WIPRO	✓
✓	zx:za:ee	BYE	BYE	✓

Passed all tests! ✓

You are provided a string of words and a 2-digit number. The two digits of the number represent the two words that are to be processed.

For example:

If the string is "Today is a Nice Day" and the 2-digit number is 41, then you are expected to process the 4th word ("Nice") and the 1st word ("Today").

The processing of each word is to be done as follows:

Extract the Middle-to-Begin part: Starting from the middle of the word, extract the characters till the beginning of the word.

Extract the Middle-to-End part: Starting from the middle of the word, extract the characters till the end of the word.

If the word to be processed is "Nice":

Its Middle-to-Begin part will be "iN".

Its Middle-to-End part will be "ce".

So, merged together these two parts would form "iNce".

Similarly, if the word to be processed is "Today":

Its Middle-to-Begin part will be "doT".

Its Middle-to-End part will be "day".

So, merged together these two parts would form "doTday".

Note: Note that the middle letter 'd' is part of both the extracted parts. So, for words whose length is odd, the middle letter should be included in both the extracted parts.

Expected output:

The expected output is a string containing both the processed words separated by a space "iNce doTday"

Example 1:

input1 = "Today is a Nice Day"

input2 = 41

output = "iNce doTday"

Example 2:

input1 = "Fruits like Mango and Apple are common but Grapes are rare"

input2 = 39

output = "naMngo arGpes"

Note: The input string input1 will contain only alphabets and a single space character separating each word in the string.

Note: The input string input1 will NOT contain any other special characters.

Note: The input number input2 will always be a 2-digit number (>=11 and <=99). One of its digits will never be 0. Both the digits of the number will always point to a valid word in the input1 string.

For example:

Input	Result
Today is a Nice Day 41	iNce doTday
Fruits like Mango and Apple are common but Grapes are rare 39	naMngo arGpes

```
import java.util.Scanner;
import java.util.Arrays;
import java.lang.String;
class prog {
   public static void main(String[] args) {
      Scanner o=new Scanner(System.in);
      String s=o.nextLine();
      int n=o.nextInt();
      String result = processWords(s,n);
      System.out.println(result);
   }
   public static String processWords(String input1, int input2) {
      String[] words = input1.split(" ");
      int firstIndex = (input2 / 10) - 1;
      int secondIndex = (input2 % 10) - 1;
      String firstWordProcessed = processWord(words[firstIndex]);
      String secondWordProcessed = processWord(words[secondIndex]);
      return firstWordProcessed + " " + secondWordProcessed;
   }
   public static String processWord(String word) {
      int length = word.length();
      int mid = length / 2;
      String l, f;
      if (length % 2 == 0) {
          f=word.substring(0,mid);
         f= new StringBuilder(f).reverse().toString();
         l= word.substring(mid);
         return f+l ;
      } else {
         f = word.substring(0, mid + 1);
         f= new StringBuilder(f).reverse().toString();
```

```
            l= word.substring(mid);
        }
        return f+l;
    }
}
```

	Input	Expected	Got	
✓	Today is a Nice Day 41	iNce doTday	iNce doTday	✓
✓	Fruits like Mango and Apple are common but Grapes are rare 39	naMngo arGpes	naMngo arGpes	✓

Passed all tests! ✓

Given 2 strings input1 & input2.

- Concatenate both the strings.
- Remove duplicate alphabets & white spaces.
- Arrange the alphabets in descending order.

Assumption 1:

There will either be alphabets, white spaces or null in both the inputs.

Assumption 2:

Both inputs will be in lower case.

Example 1:

Input 1: apple

Input 2: orange

Output: rponlgea

Example 2:

Input 1: fruits

Input 2: are good

Output: utsroigfeda

Example 3:

Input 1: ""

Input 2: ""

Output: null

For example:

Test	Input	Result
1	apple orange	rponlgea
2	fruits are good	utsroigfeda

```
import java.util.*;
public class s {
    public static String solve(String a, String b) {
        if ((a == null || a.trim().isEmpty()) && (b == null || b.trim().isEmpty())) return "null";
        String combined = a + b;
        Set<Character> uniqueChars = new HashSet<>();
        for (char c : combined.toCharArray()) {
            if (Character.isAlphabetic(c)) {
                uniqueChars.add(c);
            }
        }
        char[] charArray = new char[uniqueChars.size()];
```

```
        int i = 0;
        for (char c : uniqueChars) {
            charArray[i++] = c;
        }
        Arrays.sort(charArray);
        return new StringBuilder(new String(charArray)).reverse().toString();
    }
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        String input1 = sc.nextLine();
        String input2 = sc.nextLine();
        System.out.println(solve(input1, input2));
    }
}
```

	Test	Input	Expected	Got	
✓	1	apple orange	rponlgea	rponlgea	✓
✓	2	fruits are good	utsroigfeda	utsroigfeda	✓
✓	3		null	null	✓

Passed all tests! ✓