Set and Map Exercises

1. Given a std::vector of elements, print all the unique elements of the array in descending order.

2. Given 1 std::vector of words and a **std::unordered_set** of words, print the common words between both containers.

3. **std::set<int> union(const std::set<int>& s1, const std::set<int>& s2)**: Receives 2 sets and builds a resulting set that corresponds to the union of s1 and s2. You cannot use any function from <algorithm>

4. Given a string *s* consisting of lowercase English letters, return the first letter to appear twice. A letter *a* appears twice before another letter *b* if the second occurrence of *a* is before the second occurrence of *b*. s will contain at least one letter that appears twice.
   Example 1:
   Input: s = "abccbaacz"
   Output: "c"
   Example 2:
   Input: s = "abcdd"
   Output: "d"

5. A **pangram** is a sentence where every letter of the English alphabet appears at least once. Given a string sentence containing only lowercase English letters, return true if the sentence is a pangram, or false otherwise.
   Example 1:
   Input: sentence = "thequickbrownfoxjumpsoverthelazydog"
   Output: true
   Example 2:
   Input: sentence = "manish"
   Output: false

6. You're given a string of jewels representing the types of stones that are jewels, and stones representing the stones you have. Each character in stones is a type of stone you have. You want to know how many of the stones you have are also jewels.
   Letters are case sensitive, so "a" is considered a different type of stone from "A".

   Example 1:
   Input: jewels = "aA", stones = "aAAbbbb"
   Output: 3

   Example 2:
   Input: jewels = "z", stones = "ZZ"
   Output: 0

7. Given a string s, return true if s is a good string, or false otherwise.
   A string s is good if all the characters that appear in s have the same number of occurrences (i.e., the same frequency).

Example 1:
Input: s = "abacbc"
Output: true

Example 2:
Input: s = "aaabb"
Output: false

8. This task is very simple. Given a string S of length n and q queries each query is on the format $i\ j\ k$ which means sort the substring consisting of the characters from i to j in non-decreasing order if $k = 1$ or in non-increasing order if $k = 0$.

   Output the final string after applying the queries.

   ### Input

   The first line will contain two integers n, q ($1 \leq n \leq 10^5$, $0 \leq q \leq 50\,000$), the length of the string and the number of queries respectively.

   Next line contains a string S itself. It contains only lowercase English letters.

   Next q lines will contain three integers each i, j, k ($1 \leq i \leq j \leq n$, $k \in \{0, 1\}$).

   ### Output

   Output one line, the string S after applying the queries.

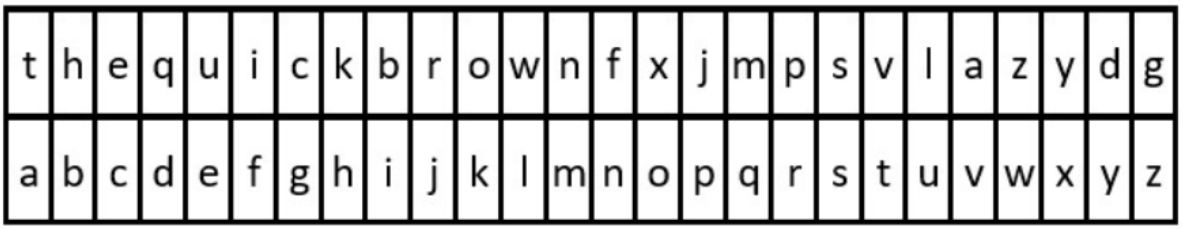
   Example 1:
   Input:
   10 5
   abacdabcda
   7 10 0
   5 8 1
   1 4 0
   3 6 0
   7 10 1
   Output:
   cbcaaaabdd

   $$abacda\mathbf{bcda} \rightarrow abacda\mathbf{dcba}$$
   $$abac\mathbf{dadc}ba \rightarrow abac\mathbf{acdd}ba$$
   $$\mathbf{abac}acddba \rightarrow \mathbf{cbaa}acddba$$
   $$cb\mathbf{aaac}ddba \rightarrow cb\mathbf{caaa}ddba$$
   $$cbcaaa\mathbf{ddba} \rightarrow cbcaaa\mathbf{abdd}$$

9. You are given the strings **key** and **message**, which represent a cipher key and a secret message, respectively. The steps to decode message are as follows:
   Use the first appearance of all 26 lowercase English letters in key as the order of the substitution table. Align the substitution table with the regular English alphabet.
   Each letter in message is then substituted using the table.
   Spaces ' ' are transformed to themselves.

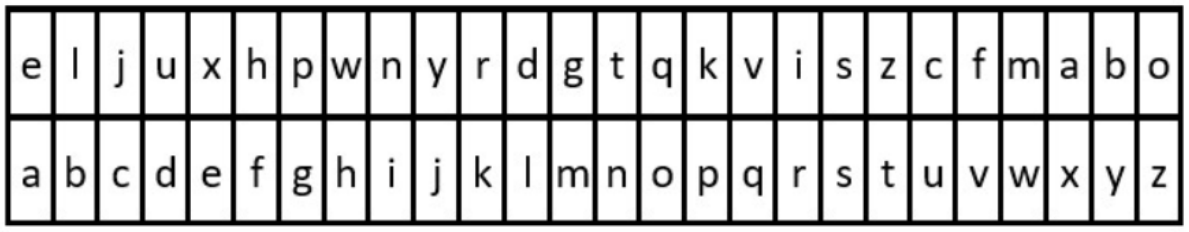
   For example, given key = "happy boy" (actual key would have at least one instance of each letter in the alphabet), we have the partial substitution table of ('h' -> 'a', 'a' -> 'b', 'p' -> 'c', 'y' -> 'd', 'b' -> 'e', 'o' -> 'f').
   Return the decoded message.

Example 1:




```
Input: key = "the quick brown fox jumps over the lazy dog", message = "vkbs bs
t suepuv"
Output: "this is a secret"
Explanation: The diagram above shows the substitution table.
It is obtained by taking the first appearance of each letter in "the quick
brown fox jumps over the lazy dog".
```

Example 2:




```
Input: key = "eljuxhpwnyrdgtqkviszcfmabo", message = "zwx hnfx lqantp mnoeius
ycgk vcnjrdb"
Output: "the five boxing wizards jump quickly"
Explanation: The diagram above shows the substitution table.
It is obtained by taking the first appearance of each letter in
"eljuxhpwnyrdgtqkviszcfmabo".
```

10. Given two strings **s** and **t**, determine if they are **isomorphic**.
    Two strings s and t are isomorphic if the characters in s can be replaced to get t. All occurrences of a character must be replaced with another character while preserving the order of characters. No two characters may map to the same character, but a character may map to itself.
    Example 1:
    Input: s = "egg", t = "add"
    Output: true

    Example 2:
    Input: s = "foo", t = "bar"
    Output: false

    Example 3:
    Input: s = "paper", t = "title"
    Output: true

11. You are given two 2D integer arrays, **items1** and **items2**, representing two sets of items. Each array items has the following properties:
    a. **items[i] = [valuei, weighti]** where **valuei** represents the value and **weighti** represents the weight of the ith item.
    b. The value of each item in items is **unique**.

    Return a 2D integer array ret where **ret[i] = [valuei, weighti]**, with **weighti** being the sum of weights of all items with value **valuei**.
    Note: **ret** should be returned in ascending order by value.

    Example 1:
    Input: items1 = [[1,1],[4,5],[3,8]], items2 = [[3,1],[1,5]]
    Output: [[1,6],[3,9],[4,5]]

    Example 2:
    Input: items1 = [[1,1],[3,2],[2,3]], items2 = [[2,1],[3,2],[1,3]]
    Output: [[1,4],[2,4],[3,4]]