1st feb 2023

FIRST:-

Isomorphic Strings : : Easy

Given two strings '**str1**' and '**str2**', check if these two strings are isomorphic to each other.

Two strings str1 and str2 are called isomorphic if there is a one to one mapping possible for every character of str1 to every character of str2 while **preserving the order**.

Note: All occurrences of every character in str1 should map to the same character in str2

Example 1:

```
Input:
str1 = aab
str2 = xxy
Output: 1
Explanation: There are two different
charactersin aab and xxy, i.e a and b
with frequency 2and 1 respectively.
```

Example 2:

```
Input:
str1 = aab
str2 = xyz
Output: 0
Explanation: There are two different
charactersin aab but there are three
different charactersin xyz. So there
won't be one to one mapping between
str1 and str2.
```

Your Task:
You don't need to read input or print anything.Your task is to complete the function **areIsomorphic**() which takes the string **str1** and string **str2** as input parameter and check if two strings are isomorphic. The function returns **true** if strings are isomorphic else it returns **false**.

Expected Time Complexity: O(|str1|+|str2|).
Expected Auxiliary Space: O(Number of different characters).
Note: |s| represents the length of string s.

Constraints:
1 <= |str1|, |str2| <= $2*10^4$

CODE SECTION:-

```
bool areIsomorphic(string str1, string str2)
    {
        // Your code here
        if(str1.length()!=str2.length()) return false;

        unordered_map<char,char>m1,m2;


        for(int i=0;i<str1.length()+1;i++){

             char s=str1[i];
             m1[s]=str2[i];
             m2[str2[i]]=str1[i];

         }

        for(int i=0;i<str1.size();i++){

            char p=str1[i];
            if(m1[str1[i]]!=str2[i])
            return false;

            if(m2[str2[i]]!=str1[i]) return false;
        }

        return true;
    }
```

-: DONE FOR THE DAY :-