Ref : LTTS/HR/ET/2022-23/11736

Name : Kamireddy Sindhuri

PROJECT

Problem:

Shil is your new boss and he likes palindromes very much. Palindrome is a string that can be read the same way in either direction, from the left to the right and from the right to the left. (ex. madam, aabaa, racecar)

Given a string S , beautiful Palindrome is a lexicographical minimum palindrome that can be formed by rearranging all the characters of string S. In order to please your boss, find a beautiful Palindrome that can be formed with help of string S.

String x is lexicographically less than string y, if either x is a prefix of y (and $x \neq y$), or there exists such i ($1 \leq i \leq \min(|x|, |y|)$), that $x_i < y_i$, and for any j ($1 \leq j < i$) $x_j = y_j$. Here $|a|$ denotes the length of the string a. The lexicographic comparison of strings is implemented by operator < in modern programming languages.

Input:

Only line of input contains string S. All the letters of this string will be in lower letters('a' - 'z').

Output:

Output lexicographical minimum Palindrome that can be formed by rearranging all the letters of string S. If no such Palindrome exist for given input, print -1.

Constraints:

$1 \leq |S| \leq 100000$

Sample Input:

aabcc

Sample Output:

Acbca

CODE:

```
#include<stdio.h>
#include<string.h>
int main()
{
  char str[100000];
  scanf("%s", str);
  int sum[26]={0};
  for(int i=0; i<strlen(str); i++)
    {
switch(str[i])
       {
           case 'a': sum[0]++;
               break;
           case 'b': sum[1]++;
               break;
           case 'c': sum[2]++;
               break;
           case 'd': sum[3]++;
               break;
           case 'e': sum[4]++;
               break;
           case 'f': sum[5]++;
               break;
           case 'g': sum[6]++;
               break;
           case 'h': sum[7]++;
               break;
           case 'i': sum[8]++;
               break;
          case 'j': sum[9]++;
```

```
    break;
case 'k': sum[10]++;
    break;
case 'l': sum[11]++;
    break;
case 'm': sum[12]++;
    break;
case 'n': sum[13]++;
    break;
case 'o': sum[14]++;
    break;
case 'p': sum[15]++;
    break;
case 'q': sum[16]++;
    break;
case 'r': sum[17]++;
    break;
case 's': sum[18]++;
    break;
case 't': sum[19]++;
    break;
case 'u': sum[20]++;
    break;
case 'v': sum[21]++;
    break;
case 'w': sum[22]++;
    break;
case 'x': sum[23]++;
    break;
case 'y': sum[24]++;
    break;
```

```
        case 'z': sum[25]++;
             break;
    }
}
int len=strlen(str);
int j=0;
int od=1;
for(int i=0 ; i<26 ; i++)
{
    if(sum[i]>0)
    {
        if(sum[i]%2==0)
        {
            while(sum[i]>0)
            {
                str[j]= i+97;
                str[len-1]= i+97;
                j++;
                len--;
                sum[i]-=2;
            }
        }
        else
        {
            if(sum[i]==1)
                str[(strlen(str)/2)]=i+97;
            else
            {
                str[(strlen(str)/2)]=i+97;
                 sum[i]--;
                 while(sum[i]>0)
```

```
            {
                str[j]= i+97;
                str[len-1]= i+97;
                j++;
                len--;
                sum[i]-=2;
             }
           }
        }
    }
 }
 len=strlen(str);
 int count=0;
for(int i=0; i<strlen(str)/2; i++)
 {
    if(str[i]==str[len-1])
    {
       count++;
       len--;
    }
    else
    {
 printf("-1");
       return 0;
    }
 }
 if(count==strlen(str)/2)
    printf("%s", str);
 else
    printf("-1");
}
```