A. Petya and Strings

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Little Petya loves presents. His mum bought him two strings of the same size for his birthday. The strings consist of uppercase and lowercase Latin letters. Now Petya wants to compare those two strings lexicographically. The letters' case does not matter, that is an uppercase letter is considered equivalent to the corresponding lowercase letter. Help Petya perform the comparison.

Input

Each of the first two lines contains a bought string. The strings' lengths range from 1 to 100 inclusive. It is guaranteed that the strings are of the same length and also consist of uppercase and lowercase Latin letters.

Output

If the first string is less than the second one, print "-1". If the second string is less than the first one, print "1". If the strings are equal, print "0". Note that the letters' case is not taken into consideration when the strings are compared.

Examples

input Copy
```
aaaa
aaaA
```
output Copy
```
0
```

input Copy
```
abs
Abz
```
output Copy
```
-1
```

input Copy
```
abcdefg
AbCdEfF
```
output Copy
```
1
```

Note

If you want more formal information about the lexicographical order (also known as the "dictionary order" or "alphabetical order"), you can visit the following site:

- `http://en.wikipedia.org/wiki/Lexicographical_order`

```cpp
/*
Q1 - Petya and Strings
Problem Link: https://codeforces.com/problemset/problem/112/A
*/

#include <bits/stdc++.h>
using namespace std;

int main() {
    string s,t; cin>>s>>t;
    int n = s.size();
    for(int i=0 ; i<n ; i++) {
        if(isupper(s[i])) s[i] = tolower(s[i]);
        if(isupper(t[i])) t[i] = tolower(t[i]);

        // Now both s[i] and t[i] are lowercase
        if(s[i] < t[i]) {
            cout<<-1<<endl;
            return 0;
        }
        if(s[i] > t[i]) {
            cout<<1<<endl;
            return 0;
        }
    }
    cout<<0<<endl;
    return 0;
}
```

344. Reverse String

Easy Topics Companies Hint

Write a function that reverses a string. The input string is given as an array of characters `s`.

You must do this by modifying the input array in-place with `O(1)` extra memory.

Example 1:

```
Input: s = ["h","e","l","l","o"]
Output: ["o","l","l","e","h"]
```

Example 2:

```
Input: s = ["H","a","n","n","a","h"]
Output: ["h","a","n","n","a","H"]
```

Constraints:

- $1 <= s.length <= 10^5$
- `s[i]` is a printable ascii character.

```cpp
/*
Q2 - Reverse String
Problem Link: https://leetcode.com/problems/reverse-string/
*/

#include <bits/stdc++.h>
using namespace std;

// Solution Code
class Solution {
public:
    void reverseString(vector<char>& s) {
        int n = s.size();
        for(int i=0 ; i<(n/2) ; i++) {
            // swap ith and (n-i-1)th character
            char temp = s[i];
            s[i] = s[n-i-1];
            s[n-i-1] = temp;
        }
    }
};

//Driver Code
int main() {
    int n; cin>>n;
    vector<char> s(n);
    for(int i=0 ; i<n ; i++) cin>>s[i];
    Solution obj;
    obj.reverseString(s);
    for(int i=0 ; i<n ; i++) cout<<s[i]<<" ";
    return 0;
}
```

B. Two-gram

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Two-gram is an ordered pair (i.e. string of length two) of capital Latin letters. For example, "AZ", "AA", "ZA" — three distinct two-grams.

You are given a string s consisting of n capital Latin letters. Your task is to find **any** two-gram contained in the given string **as a substring** (i.e. two consecutive characters of the string) maximal number of times. For example, for string s = "BBAABBBA" the answer is two-gram "BB", which contained in s three times. In other words, find any most frequent two-gram.

Note that occurrences of the two-gram can overlap with each other.

Input

The first line of the input contains integer number n $(2 \le n \le 100)$ — the length of string s. The second line of the input contains the string s consisting of n capital Latin letters.

Output

Print the only line containing exactly two capital Latin letters — **any** two-gram contained in the given string s **as a substring** (i.e. two consecutive characters of the string) maximal number of times.

Examples

input

```
7
ABACABA
```

output

```
AB
```

input

```
5
ZZZAA
```

output

```
ZZ
```

Note

In the first example "BA" is also valid answer.

In the second example the only two-gram "ZZ" can be printed because it contained in the string "ZZZAA" two times.

```cpp
/*
Q3 - Two-gram
Problem Link: https://codeforces.com/problemset/problem/977/B
*/

#include <bits/stdc++.h>
using namespace std;

int main() {
    int n; cin>>n;
    string s; cin>>s;

    // variables to store final answer
    string maxStr;
    int maxFreq = 0;

    for(int i=0 ; i<n-1 ; i++) {
        // current string consistes of characters s[i] and s[i+1];
        int currFreq = 0;
        for(int j=0 ; j<n-1 ; j++) {
            if(s[i]==s[j] && s[i+1]==s[j+1]) {
                currFreq++;
            }
        }

        if(currFreq > maxFreq) {
            maxFreq = currFreq;
            maxStr.clear();
            maxStr.push_back(s[i]);
            maxStr.push_back(s[i+1]);
        }
    }

    cout<<maxStr<<endl;

    return 0;
}
```

9. Palindrome Number

Easy   Topics   Companies   Hint

Given an integer `x`, return `true` *if* `x` *is a* ***palindrome****, and* `false` *otherwise.*

Example 1:

```
Input: x = 121
Output: true
Explanation: 121 reads as 121 from left to right and from right to left.
```

Example 2:

```
Input: x = -121
Output: false
Explanation: From left to right, it reads -121. From right to left, it
becomes 121-. Therefore it is not a palindrome.
```

Example 3:

```
Input: x = 10
Output: false
Explanation: Reads 01 from right to left. Therefore it is not a palindrome.
```

Constraints:

- $-2^{31} <= x <= 2^{31} - 1$

Follow up: Could you solve it without converting the integer to a string?

```cpp
/*
Q4 - Palindrome Number
Problem link: https://leetcode.com/problems/palindrome-number/
*/

#include <bits/stdc++.h>
using namespace std;

// Solution code
class Solution {
public:
    bool isPalindrome(int x) {
        if(x<0) return false;
        // declare rev as long because reverse of 2^31 will not fit in integer range.
        long int rev = 0;
        int copy = x;
        while(copy) {
            int dig = copy%10;
            rev = rev*10 + dig;
            copy /= 10;
        }
        return (rev == x);
    }
};

//Driver code
int main() {
    int n; cin>>n;
    Solution obj;
    if(obj.isPalindrome(n)) cout<<"true"<<endl;
    else cout<<"false";
    return 0;
}
```

125. Valid Palindrome

Easy Topics Companies

A phrase is a **palindrome** if, after converting all uppercase letters into lowercase letters and removing all non-alphanumeric characters, it reads the same forward and backward. Alphanumeric characters include letters and numbers.

Given a string `s`, return `true` *if it is a **palindrome**, or* `false` *otherwise.*

Example 1:

```
Input: s = "A man, a plan, a canal: Panama"
Output: true
Explanation: "amanaplanacanalpanama" is a palindrome.
```

Example 2:

```
Input: s = "race a car"
Output: false
Explanation: "raceacar" is not a palindrome.
```

Example 3:

```
Input: s = " "
Output: true
Explanation: s is an empty string "" after removing non-alphanumeric
characters.
Since an empty string reads the same forward and backward, it is a
palindrome.
```

Constraints:

- `1 <= s.length <= 2 * 10`^{5}
- `s` consists only of printable ASCII characters.

```cpp
/*
Q5 - Valid Palindrome
Problem Link: https://leetcode.com/problems/valid-palindrome/
*/

#include <bits/stdc++.h>
using namespace std;

// Solution Code
class Solution {
private:
    bool isAlphanumeric(char c) {
        if(c>='0' && c<='9') return true;
        if(c>='a' && c<='z') return true;
        if(c>='A' && c<='Z') return true;
        return false;
    }
public:
    bool isPalindrome(string s) {
        string valStr;
        for(int i=0 ; i<(int)s.size() ; i++) {
            if(isAlphanumeric(s[i])) {
                if(isupper(s[i])) s[i] = tolower(s[i]);
                valStr.push_back(s[i]);
            }
        }
        string revStr = valStr;
        // inbuilt function to reverse a string
        reverse(revStr.begin(), revStr.end());
        return (revStr == valStr);
    }
};

// Driver Code
int main() {
    string s;
    getline(cin, s);
    Solution obj;
    if(obj.isPalindrome(s)) cout<<"true"<<endl;
    else cout<<"false"<<endl;
    return 0;
}
```

680. Valid Palindrome II

Easy Topics Companies

Given a string `s`, return `true` *if the* `s` *can be palindrome after deleting* ***at most one*** *character from it*.

Example 1:

```
Input: s = "aba"
Output: true
```

Example 2:

```
Input: s = "abca"
Output: true
Explanation: You could delete the character 'c'.
```

Example 3:

```
Input: s = "abc"
Output: false
```

Constraints:

- $1 <= s.length <= 10^5$
- `s` consists of lowercase English letters.

```cpp
/*
Q6 - Valid Palindrome II
Problem Link: https://leetcode.com/problems/valid-palindrome-ii/description/
*/

#include <bits/stdc++.h>
using namespace std;

//Solution Code
class Solution {
private:
    bool isPalindrome(string &s, int i, int j) {
        while(i<j) {
            if(s[i] == s[j]) {
                i++;
                j--;
            } else return false;
        }
        return true;
    }
public:
    bool validPalindrome(string s) {
        int n = s.size();
        int i=0;
        int j=n-1;
        while(i<j) {
            if(s[i]!=s[j]) {
                //either delete s[i] or s[j]
                if(isPalindrome(s, i, j-1) || isPalindrome(s, i+1, j)) {
                    return true;
                } else return false;
            } else {
                i++;
                j--;
            }
        }
        return true;
    }
};

//Driver code
int main() {
    string s; cin>>s;
    Solution obj;
    if(obj.validPalindrome(s)) cout<<"true"<<endl;
    else cout<<"false"<<endl;
    return 0;
}
```