[A - Pangram](https://vjudge.net/problem/CodeForces-520A)

A word or a sentence in some language is called a *pangram* if all the characters of the alphabet of this language appear in it *at least once*. Pangrams are often used to demonstrate fonts in printing or test the output devices.

You are given a string consisting of lowercase and uppercase Latin letters. Check whether this string is a pangram. We say that the string contains a letter of the Latin alphabet if this letter occurs in the string in uppercase or lowercase.

**Input**

The first line contains a single integer *n* (1 ≤ *n* ≤ 100) — the number of characters in the string.

The second line contains the string. The string consists only of uppercase and lowercase Latin letters.

**Output**

Output "YES", if the string is a pangram and "NO" otherwise.

**Examples**

**Input**

12  
toosmallword

**Output**

NO

**Input**

35  
TheQuickBrownFoxJumpsOverTheLazyDog

**Output**

YES

Pangram

In given string, we need to ensure that every character is present.  Simplest way to ensure that is to take a 1-D BOOLEAN ARRAY of size 26.

bool isPangram(string s){  
    bool arr[26];  
    for each character ch in s  
        arr[ch - 'a'] = 1 // for 'a' mark index 0, for 'b' mark index 1 and so on.  
  
    for i = [0, 25]  
        if arr[i] is false  
            return false; // found a character whose freq is 0, which means missing char  
  
    return true; // Reaching here means, all characters are present.  
}

Time Complexity:  length(s) + 26 = O(length(s))

Space Complexity:  26 (size of frequency array) = O(alphabetSize).  Note: We can call it O(1) also.

BITWISE APPROACH

A single integer has 32-bits in it.  What if 26-bits can be used as availability of each 26-characters (0th bit - 'a', 1st bit - 'b' and so on.

bool isPangram(string s){  
    int x = 0; // Binary form is all 32-bits as 0  
    for each character ch in s  
        int bitIndex = ch - 'a'; // for 'a' => 0, for 'b' => 1 and so on.  
        x = setBitInX(x, bitIndex); // x = x | (1 << bitIndex);  
  
    for i = [0, 25]  
        if isIthBitSet(x, i)==false // return (x & (1 << i)) != 0;  
            return false; // found a missing character  
  
    return true; // Reaching here means, all characters are present.  
}

Time Complexity:  length(s) + 26 = O(length(s))

Space Complexity:  1 = O(1)

#include <iostream>

#include <algorithm>

#include <string>

using namespace *std*;

bool isOnlyUppercase(*string*& in)

{

for (auto i = 0; i < in.*size*(); i++)

{

if (in[i] >= 'a' && in[i] <= 'z')

return false;

}

return true;

}

bool isFirstLowerAndAllOtherUppser(*string* &in)

{

for (auto i = 1; i < in.*size*(); i++)

{

if (in[i] >= 'a' && in[i] <= 'z')

return false;

}

if (in[0] >= 'a' && in[0] <= 'z')

{

in[0] = *toupper*(in[0]);

return true;

}

return false;

}

int main(void) {

*ios*::*sync\_with\_stdio*(0);

*cin*.*tie*(0);

*cout*.*tie*(0);

int n; *cin* >> n;

char alphabets[26] = { '\0' };

for (auto i = 0; i < n; i++)

{

char ch = '\0';

*cin* >> ch;

ch = *tolower*(ch);

alphabets[ch - 97] = ch;

}

if (n < 26)

{

*cout* << "NO\n";

return 0;

}

bool ispangram = true;

for (auto i = 0; i < 26; i++)

if (alphabets[i] == '\0')

{

ispangram = false;

*cout* << "NO\n";

break;

}

if (ispangram)

*cout* << "YES\n";

return 0;

}