



Full Name:	pravallika
Email:	pravallikajoseph20@gmail.com
Test Name:	Mock Test
Taken On:	25 Aug 2025 09:46:12 IST
Time Taken:	84 min 4 sec/ 90 min
Invited by:	Ankush
Invited on:	25 Aug 2025 09:46:00 IST
Skills Score:	
Tags Score:	<div>Algorithms280/280</div> <div>Core CS280/280</div> <div>Data Structures105/105</div> <div>Easy280/280</div> <div>LCM105/105</div> <div>Least Common Multiple105/105</div> <div>Math105/105</div> <div>Problem Solving105/105</div> <div>Strings175/175</div> <div>gcd105/105</div> <div>greatest common divisor105/105</div> <div>problem-solving280/280</div> <div>sets105/105</div>

100%
280/280

scored in **Mock Test** in 84 min 4 sec on 25 Aug 2025 09:46:12 IST

Recruiter/Team Comments:

No Comments.

Plagiarism flagged

We have marked questions with suspected plagiarism below. Please review it in detail here -

	Question Description	Time Taken	Score	Status
Q1	Palindrome Index > Coding	31 min 35 sec	105/ 105	⚠
Q2	Between Two Sets > Coding	19 min 34 sec	105/ 105	⚠
Q3	Anagram > Coding	31 min 4 sec	70/ 70	⚠

QUESTION 1



Needs Review

Score 105

Palindrome Index > Coding

Strings

Algorithms

Easy

problem-solving

Core CS

Problem Solving

QUESTION DESCRIPTION

Given a string of lowercase letters in the range `ascii[a-z]`, determine the index of a character that can be removed to make the string a **palindrome**. There may be more than one solution, but any will do. If the word is already a palindrome or there is no solution, return `-1`. Otherwise, return the index of a character to remove.

Example

 $s = \text{"bcbc"}$

Either remove 'b' at index **0** or 'c' at index **3**.

Function Description

Complete the `palindromeIndex` function in the editor below.

`palindromeIndex` has the following parameter(s):

- string s*: a string to analyze

Returns

- int*: the index of the character to remove or **-1**

Input Format

The first line contains an integer **q** , the number of queries.

Each of the next **q** lines contains a query string **s** .

Constraints

- $1 \leq q \leq 20$
- $1 \leq \text{length of } s \leq 10^5 + 5$
- All characters are in the range `ascii[a-z]`.

Sample Input

STDIN	Function
-----	-----
3	$q = 3$
aaab	$s = \text{'aaab'}$ (first query)
baa	$s = \text{'baa'}$ (second query)
aaa	$s = \text{'aaa'}$ (third query)

Sample Output

```
3
0
-1
```

Explanation

Query 1: "aaab"

Removing 'b' at index **3** results in a palindrome, so return **3**.

Query 2: "baa"

Removing 'b' at index **0** results in a palindrome, so return **0**.

Query 3: "aaa"

This string is already a palindrome, so return **-1**. Removing any one of the characters would result in a palindrome, but this test comes first.

Note: The custom checker logic for this challenge is available [here](#).

CANDIDATE ANSWER

Language used: C

```

1 int isPalindrome(char *s,int l, int r){
2     while(l<r){
3         if(s[l] != s[r]){
4             return 0;
5         }
6         l++;
7         r--;
8     }
9     return 1;
10 }
11 int palindromeIndex(char* s) {
12     int n = strlen(s);
13     int l = 0, r = n-1;
14     while(l<r){
15         if(s[l] != s[r]){
16             if(isPalindrome(s,l+1,r)){
17                 return l;
18             }
19             if(isPalindrome(s,l,r-1)){
20                 return r;
21             }
22             return -1;
23         }
24         l++;
25         r--;
26     }
27     return -1;
28 }
29

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	✔ Success	0	0.008 sec	7.13 KB
Testcase 2	Medium	Hidden case	✔ Success	5	0.0079 sec	7 KB
Testcase 3	Medium	Hidden case	✔ Success	5	0.0067 sec	7 KB
Testcase 4	Medium	Hidden case	✔ Success	5	0.0082 sec	6.88 KB
Testcase 5	Medium	Hidden case	✔ Success	5	0.0066 sec	7.25 KB
Testcase 6	Medium	Hidden case	✔ Success	5	0.0096 sec	7.38 KB
Testcase 7	Medium	Hidden case	✔ Success	5	0.009 sec	7.13 KB
Testcase 8	Medium	Hidden case	✔ Success	5	0.0086 sec	7.75 KB
Testcase 9	Hard	Hidden case	✔ Success	10	0.0093 sec	7.5 KB
Testcase 10	Hard	Hidden case	✔ Success	10	0.0073 sec	7.13 KB
Testcase 11	Hard	Hidden case	✔ Success	10	0.0074 sec	7.13 KB
Testcase 12	Hard	Hidden case	✔ Success	10	0.0112 sec	6.88 KB
Testcase 13	Hard	Hidden case	✔ Success	10	0.0109 sec	7.25 KB
Testcase 14	Hard	Hidden case	✔ Success	10	0.01 sec	7.38 KB
Testcase 15	Hard	Hidden case	✔ Success	10	0.0082 sec	7.38 KB

QUESTION 2



Needs Review

Score 105

Between Two Sets > Coding

Math

Algorithms

Easy

gcd

Data Structures

LCM

sets

problem-solving

Core CS

greatest common divisor

Least Common Multiple

QUESTION DESCRIPTION

There will be two arrays of integers. Determine all integers that satisfy the following two conditions:

1. The elements of the first array are all factors of the integer being considered
2. The integer being considered is a factor of all elements of the second array

These numbers are referred to as being *between* the two arrays. Determine how many such numbers exist.

Example

 $a = [2, 6]$
 $b = [24, 36]$

There are two numbers between the arrays: **6** and **12**.

 $6\%2 = 0, 6\%6 = 0, 24\%6 = 0$ and $36\%6 = 0$ for the first value.

 $12\%2 = 0, 12\%6 = 0$ and $24\%12 = 0, 36\%12 = 0$ for the second value. Return **2**.

Function Description

Complete the `getTotalX` function in the editor below. It should return the number of integers that are between the sets.

`getTotalX` has the following parameter(s):

- `int a[n]`: an array of integers
- `int b[m]`: an array of integers

Returns

- `int`: the number of integers that are between the sets

Input Format

The first line contains two space-separated integers, n and m , the number of elements in arrays a and b .

The second line contains n distinct space-separated integers $a[i]$ where $0 \leq i < n$.

The third line contains m distinct space-separated integers $b[j]$ where $0 \leq j < m$.

Constraints

- $1 \leq n, m \leq 10$
- $1 \leq a[i] \leq 100$
- $1 \leq b[j] \leq 100$

Sample Input

```
2 3
2 4
16 32 96
```

Sample Output

```
3
```

Explanation

2 and 4 divide evenly into 4, 8, 12 and 16.

4, 8 and 16 divide evenly into 16, 32, 96.

4, 8 and 16 are the only three numbers for which each element of a is a factor and each is a factor of all elements of b .

CANDIDATE ANSWER

Language used: C

```

1  int gcd(int a, int b){
2      while(b != 0){
3          int temp = b;
4          b = a%b;
5          a =temp;
6      }
7      return a;
8  }
9  int lcm(int a, int b){
10     return (a*b)/gcd(a,b);
11 }
12 int getTotalX(int a_count, int* a, int b_count, int* b) {
13     int l = a[0];
14     for(int i = 1; i<a_count; i++){
15         l = lcm(l,a[i]);
16     }
17     int g = b[0];
18     for(int i = 1; i<b_count;i++){
19         g = gcd(g,b[i]);
20     }
21     int count = 0;
22     for(int x=1;x <= g; x += 1){
23         if(g % x == 0){
24             count++;
25         }
26     }
27     return count;
28 }
29

```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	✔ Success	0	0.0101 sec	7.25 KB
Testcase 2	Easy	Hidden case	✔ Success	15	0.0067 sec	7.38 KB
Testcase 3	Easy	Hidden case	✔ Success	15	0.0069 sec	7.13 KB
Testcase 4	Easy	Hidden case	✔ Success	15	0.0113 sec	7 KB
Testcase 5	Easy	Hidden case	✔ Success	15	0.0074 sec	6.88 KB
Testcase 6	Easy	Hidden case	✔ Success	15	0.0114 sec	7 KB
Testcase 7	Easy	Hidden case	✔ Success	15	0.0074 sec	6.88 KB
Testcase 8	Easy	Hidden case	✔ Success	15	0.0071 sec	7.25 KB
Testcase 9	Easy	Sample case	✔ Success	0	0.0065 sec	7 KB

No Comments

QUESTION 3



Needs Review

Anagram > Coding

Strings

Algorithms

Easy

problem-solving

Core CS

QUESTION DESCRIPTION

Two words are *anagrams* of one another if their letters can be rearranged to form the other word.

Given a string, split it into two contiguous substrings of equal length. Determine the minimum number of characters to change to make the two substrings into anagrams of one another.

Example

$s = \text{abccde}$

Break s into two parts: 'abc' and 'cde'. Note that all letters have been used, the substrings are contiguous and their lengths are equal. Now you can change 'a' and 'b' in the first substring to 'd' and 'e' to have 'dec' and 'cde' which are anagrams. Two changes were necessary.

Function Description

Complete the *anagram* function in the editor below.

anagram has the following parameter(s):

- *string s*: a string

Returns

- *int*: the minimum number of characters to change or -1.

Input Format

The first line will contain an integer, q , the number of test cases.

Each test case will contain a string s .

Constraints

- $1 \leq q \leq 100$
- $1 \leq |s| \leq 10^4$
- s consists only of characters in the range `ascii[a-z]`.

Sample Input

```
6
aaabbb
ab
abc
mnop
xyyx
xaxbbbx
```

Sample Output

```
3
1
-1
2
0
1
```

Explanation

Test Case #01: We split s into two strings $S1 = \text{'aaa'}$ and $S2 = \text{'bbb'}$. We have to replace all three characters from the first string with 'b' to make the strings anagrams.

Test Case #02: You have to replace 'a' with 'b', which will generate "bb".

Test Case #03: It is not possible for two strings of unequal length to be anagrams of one another.

Test Case #04: We have to replace both the characters of first string ("mn") to make it an anagram of the other one.















Test Case #05: $S1$ and $S2$ are already anagrams of one another.

Test Case #06: Here S1 = "xaxb" and S2 = "bbxx". You must replace 'a' from S1 with 'b' so that S1 = "xbxb".

CANDIDATE ANSWER

Language used: C

```
1 int anagram(char* s) {
2     int len = strlen(s);
3     if( len % 2 != 0){
4         return -1;
5     }
6     int half = len/2;
7     int freq[26]={0};
8     for(int i = 0; i<half;i++){
9         freq[s[i]-'a']++;
10    }
11    for(int i = half;i<len;i++){
12        freq[s[i]-'a']--;
13    }
14    int changes = 0;
15    for(int i =0; i<26; i++){
16        if(freq[i]>0){
17            changes += freq[i];
18        }
19    }
20    return changes;
21 }
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Hidden case	 Success	5	0.0071 sec	6.88 KB
Testcase 2	Easy	Hidden case	 Success	5	0.0078 sec	7 KB
Testcase 3	Easy	Hidden case	 Success	5	0.0077 sec	7 KB
Testcase 4	Easy	Hidden case	 Success	5	0.0085 sec	7 KB
Testcase 5	Easy	Hidden case	 Success	5	0.0096 sec	7 KB
Testcase 6	Easy	Hidden case	 Success	5	0.0111 sec	7.75 KB
Testcase 7	Easy	Hidden case	 Success	5	0.0136 sec	7.88 KB
Testcase 8	Easy	Hidden case	 Success	5	0.0175 sec	7.88 KB
Testcase 9	Easy	Hidden case	 Success	5	0.0143 sec	7.38 KB
Testcase 10	Easy	Hidden case	 Success	5	0.0237 sec	7.5 KB
Testcase 11	Easy	Hidden case	 Success	5	0.0134 sec	7.75 KB
Testcase 12	Easy	Hidden case	 Success	5	0.0181 sec	7.75 KB
Testcase 13	Easy	Hidden case	 Success	5	0.0097 sec	7.75 KB
Testcase 14	Easy	Hidden case	 Success	5	0.0101 sec	7.75 KB
Testcase 15	Easy	Sample case	 Success	0	0.008 sec	7 KB
Testcase 16	Easy	Sample case	 Success	0	0.0122 sec	7 KB

No Comments