Full Name:	Aman Kumar	
Email:	amankumar52819@gmail.com	
Test Name:	**Mock Test**	
Taken On:	20 Aug 2025 11:16:52 IST	
Time Taken:	26 min 9 sec/ 90 min	
Linkedin:	http://linkedin.com/in/aman-kumar-a19b23222	
Invited by:	Ankush	
Invited on:	20 Aug 2025 11:16:07 IST	
Skills Score:		

100%

280/280

scored in **Mock Test** in 26 min 9 sec on 20 Aug 2025 11:16:52 IST

Tags Score:

Tag	Score
Algorithms	280/280
Core CS	280/280
Data Structures	105/105
Easy	280/280
LCM	105/105
Least Common Multiple	105/105
Math	105/105
Problem Solving	105/105
Strings	175/175
gcd	105/105
greatest common divisor	105/105
problem-solving	280/280
sets	105/105

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**	9 min 16 sec	105/ 105	⚠
Q2	**Between Two Sets** > **Coding**	11 min 37 sec	105/ 105	⚠
Q3	**Anagram** > **Coding**	5 min 2 sec	70/ 70	⚠

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 = $ "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

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

Sample Input

```
STDIN    Function
-----    --------
3        q = 3
aaab     s = 'aaab' (first query)
baa      s = 'baa'  (second query)
aaa      s = '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.

Language used: **C**

```c
/*
 * Complete the 'palindromeIndex' function below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts STRING s as parameter.
 */

bool isPalindrome(char *s, int l, int r){
    while(l<r){
        if(s[l] != s[r])return false;
        l++;
        r--;
    }
    return true;
}

int palindromeIndex(char* s) {

    int n = strlen(s);
    int l=0, r=n-1;

    while(l<r){
        if(s[l] == s[r]){
            l++;
            r--;
        }else{
            if(isPalindrome(s, l+1, r)) return l;
            if(isPalindrome(s, l, r-1)) return r;
            return -1;
        }
    }
    return -1;
}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	Success	0	0.0122 sec	7.38 KB
Testcase 2	Medium	Hidden case	Success	5	0.0073 sec	7.13 KB
Testcase 3	Medium	Hidden case	Success	5	0.0136 sec	7.25 KB
Testcase 4	Medium	Hidden case	Success	5	0.0068 sec	7.38 KB
Testcase 5	Medium	Hidden case	Success	5	0.0071 sec	7.25 KB
Testcase 6	Medium	Hidden case	Success	5	0.0134 sec	7.38 KB
Testcase 7	Medium	Hidden case	Success	5	0.0098 sec	7.38 KB
Testcase 8	Medium	Hidden case	Success	5	0.0087 sec	7.75 KB
Testcase 9	Hard	Hidden case	Success	10	0.0076 sec	7.38 KB
Testcase 10	Hard	Hidden case	Success	10	0.0079 sec	7.38 KB
Testcase 11	Hard	Hidden case	Success	10	0.0106 sec	7.5 KB
Testcase 12	Hard	Hidden case	Success	10	0.0138 sec	7.38 KB

Testcase 13	Hard	Hidden case	✓ Success	10	0.0082 sec	7.38 KB
Testcase 14	Hard	Hidden case	✓ Success	10	0.0098 sec	7.5 KB
Testcase 15	Hard	Hidden case	✓ Success	10	0.0115 sec	7.5 KB

No Comments

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

Constraints

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

Sample Input

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

Sample Output

```
3
```

Explanation

4/8

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**

```c
/*
 * Complete the 'getTotalX' function below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts following parameters:
 *  1. INTEGER_ARRAY a
 *  2. INTEGER_ARRAY b
 */

int gcd(int a, int b){
    while(b != 0){
        int temp = b;
        b = a%b;
        a = temp;
    }
    return a;
}

int lcm(int a, int b){
    return (a*b) / gcd(a,b);
}

int getTotalX(int a_count, int* a, int b_count, int* b) {

    int l = a[0];
    for(int i=1; i<a_count; i++){
        l = lcm(l,a[i]);
    }

    int g = b[0];
    for(int i=1; i<b_count; i++){
        g = gcd(g, b[i]);
    }

    int count =0;
    for(int x= l ; x <= g; x+=l){
        if(g%x == 0){
            count ++;
        }
    }
    return count;
}
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED
Testcase 1	Easy	Sample case	✓ Success	0	0.015 sec	7.38 KB
Testcase 2	Easy	Hidden case	✓ Success	15	0.0136 sec	7.38 KB
Testcase 3	Easy	Hidden case	✓ Success	15	0.008 sec	7.25 KB

Testcase 4	Easy	Hidden case	✓ Success	15	0.0081 sec	7.38 KB
Testcase 5	Easy	Hidden case	✓ Success	15	0.0078 sec	7.25 KB
Testcase 6	Easy	Hidden case	✓ Success	15	0.0069 sec	7.25 KB
Testcase 7	Easy	Hidden case	✓ Success	15	0.0068 sec	7.38 KB
Testcase 8	Easy	Hidden case	✓ Success	15	0.0082 sec	7.25 KB
Testcase 9	Easy	Sample case	✓ Success	0	0.0077 sec	7.38 KB

No Comments

QUESTION 3

⚠

Needs Review

Score 70

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

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

Sample Input

```
6
aaabbb
ab
abc
mnop
xyyx
xaxbbbxx
```

Sample Output

```
3
1
```

```
-1
2
0
1
```

Explanation

Test Case #01: We split s into two strings $S1$='aaa' and $S2$='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**

```c
1
2   /*
3    * Complete the 'anagram' function below.
4    *
5    * The function is expected to return an INTEGER.
6    * The function accepts STRING s as parameter.
7    */
8
9   int anagram(char* s) {
10
11      int len = strlen(s);
12
13      if(len%2 != 0) return -1;
14
15      int half = len/2;
16      int freq[26] = {0};
17
18      for(int i=0; i<half; i++){
19          freq[s[i] - 'a']++;
20      }
21
22      for(int i= half; i<len; i++){
23          freq[s[i] - 'a']--;
24      }
25
26      int changes = 0;
27      for(int i=0; i<26; i++){
28          if(freq[i]>0) changes += freq[i];
29      }
30      return changes;
31  }
32
```

TESTCASE	DIFFICULTY	TYPE	STATUS	SCORE	TIME TAKEN	MEMORY USED

Testcase 1	Easy	Hidden case	✓ Success	5	0.0079 sec	7.25 KB
Testcase 2	Easy	Hidden case	✓ Success	5	0.007 sec	7.25 KB
Testcase 3	Easy	Hidden case	✓ Success	5	0.0072 sec	7 KB
Testcase 4	Easy	Hidden case	✓ Success	5	0.0112 sec	7.13 KB
Testcase 5	Easy	Hidden case	✓ Success	5	0.0076 sec	7.38 KB
Testcase 6	Easy	Hidden case	✓ Success	5	0.0195 sec	7.75 KB
Testcase 7	Easy	Hidden case	✓ Success	5	0.0088 sec	7.63 KB
Testcase 8	Easy	Hidden case	✓ Success	5	0.0112 sec	8.25 KB
Testcase 9	Easy	Hidden case	✓ Success	5	0.0156 sec	7.88 KB
Testcase 10	Easy	Hidden case	✓ Success	5	0.0102 sec	8.25 KB
Testcase 11	Easy	Hidden case	✓ Success	5	0.0085 sec	7.63 KB
Testcase 12	Easy	Hidden case	✓ Success	5	0.0245 sec	8.13 KB
Testcase 13	Easy	Hidden case	✓ Success	5	0.0311 sec	8.13 KB
Testcase 14	Easy	Hidden case	✓ Success	5	0.0106 sec	7.88 KB
Testcase 15	Easy	Sample case	✓ Success	0	0.0081 sec	7.25 KB
Testcase 16	Easy	Sample case	✓ Success	0	0.0163 sec	7 KB

No Comments

PDF generated at: 20 Aug 2025 06:15:06 UTC