Week 10:

ROLL NO.:240801179

Name: Lohith P Shetty

Status	Finished
Started	Sunday, 29 December 2024, 11:24 PM
Completed	Sunday, 29 December 2024, 11:25 PM
Duration	1 min 7 secs

Q1) Given a string, s, consisting of alphabets and digits, find the frequency of each digit in the given string.

Input Format

The first line contains a string, num which is the given number.

Constraints

$1 \leq \text{len(num)} \leq 1000$

All the elements of num are made of English alphabets and digits.

Output Format

Print ten space-separated integers in a single line denoting the frequency of each digit from 0 to 9.

Sample Input 0

a11472o5t6

Sample Output 0

0 2 1 0 1 1 1 1 0 0

Explanation 0

In the given string:

• 1 occurs two times.

• 2, 4, 5, 6 and 7 occur one time each.

• The remaining digits 0, 3, 8 and 9 don't occur at all.

Hint:

• Declare an array, freq of size 10 and initialize it with zeros, which will be used to count the frequencies of each of the digit occurring.

• Given a string, s, iterate through each of the character in the string. Check if the current

character is a number or not.

• If the current character is a number, increase the frequency of that position in the freq array by 1.

• Once done with the iteration over the string, s, in a new line print all the 10 frequencies starting from 0 to 9, separated by spaces.

Code:

```c
#include <stdio.h>
#include <string.h>

int main() {
    char num[1001]; // Input string (max length 1000)
    int digit_count[10] = {0}; // Array to count frequency of digits 0-9

    // Input the string
    scanf("%s", num);

    // Iterate over each character in the string
    for (int i = 0; num[i] != '\0'; i++) {
        if (num[i] >= '0' && num[i] <= '9') {
            digit_count[num[i] - '0']++; // Increment count for the respective digit
        }
    }

    // Print the frequencies of digits 0 to 9
    for (int i = 0; i < 10; i++) {
        printf("%d ", digit_count[i]);
    }

    return 0;
}
```

OUTPUT:

	Input	Expected	Got	
✓	a11472o5t6	0 2 1 0 1 1 1 1 0 0	0 2 1 0 1 1 1 1 0 0	✓
✓	lw4n88j12n1	0 2 1 0 1 0 0 0 2 0	0 2 1 0 1 0 0 0 2 0	✓
✓	1v88886l256338ar0ekk	1 1 1 2 0 1 2 0 5 0	1 1 1 2 0 1 2 0 5 0	✓

Passed all tests! ✓

Q2)

Today, Monk went for a walk in a garden. There are many trees in the garden and each tree has an English alphabet on it. While Monk was walking, he noticed that all trees with vowels on it are not in good state. He decided to take care of them. So, he asked you to tell him the count of such trees in the garden.

Note: The following letters are vowels: 'A', 'E', 'I', 'O', 'U', 'a', 'e', 'i', 'o' and 'u'.

Input Format:
The first line consists of an integer T denoting the number of test cases.
Each test case consists of only one string, each character of string denoting the alphabet (may be lowercase or uppercase) on a tree in the garden.

Output Format:
For each test case, print the count in a new line.

Constraints:
$1 \leq T \leq 10$
$1 \leq$ length of string ≤ 105

Sample Input
2
nBBZLaosnm
JHkIsnZtTL

Sample Output
2
1

Explanation
In test case 1, a and o are the only vowels. So, count=2
Brief Description: Given a string S you have to count number of vowels in the string.

Solution 1:
For each vowel, count how many times it is appearing in the string S. Final answer will the sum of frequencies of all the vowels.

Solution 2:
Iterate overall all the characters in the string S and use a counter (variable) to keep track of number of vowels in the string S. While iterating over the characters, if we encounter a vowel, we will increase the counter by 1.

Time Complexity: O(N) where N is the length of the string S. Space Complexity: O(N)

Code:

```c
#include <stdio.h>
#include <string.h>
#include <ctype.h>

int is_vowel(char ch) {
    // Convert character to lowercase
    ch = tolower(ch);
    // Check if the character is a vowel
    return (ch == 'a' || ch == 'e' || ch == 'i' || ch == 'o' || ch == 'u');
}

int main() {
    int T; // Number of test cases
    scanf("%d", &T);
    getchar(); // To consume the newline after T

    while (T--) {
        char str[100001];
        int count = 0;

        // Read the input string
        scanf("%s", str);

        // Iterate through the string and count vowels
        for (int i = 0; str[i] != '\0'; i++) {
            if (is_vowel(str[i])) {
                count++;
            }
        }

        // Print the result for the current test case
        printf("%d\n", count);
    }

    return 0;
}
```

OUTPUT:

	Input	Expected	Got	
✓	2 nBBZLaosnm JHkIsnZtTL	2 1	2 1	✓
✓	2 nBBZLaosnm JHkIsnZtTL	2 1	2 1	✓

Passed all tests! ✓

Q3) Given a sentence, s, print each word of the sentence in a new line.

Input Format

The first and only line contains a sentence, s.

Constraints

$1 \leq \text{len}(s) \leq 1000$

Output Format

Print each word of the sentence in a new line.

Sample Input

This is C

Sample Output

This

is

C

Explanation

In the given string, there are three words ["This", "is", "C"]. We have to print each of these words in a new line.

Hint

Here, once you have taken the sentence as input, we need to iterate through the input, and keep printing each character one after the other unless you encounter a space. When a space is encountered, you know that a token is complete and space indicates the start of the next token after this. So, whenever there is a space, you need to move to a new line, so that you can start printing the next token.

Code:

```c
#include <stdio.h>
#include <string.h>

int main()
{
    char str[1000];
    scanf("%[^\n]s",str);
    for(int i=0;str[i]!='\0';i++){
        if(str[i]==' ')
        printf("\n");
        else
        printf("%c",str[i]);
    }
}
```

OUTPUT:

	Input	Expected	Got	
✓	This is C	This is C	This is C	✓
✓	Learning C is fun	Learning C is fun	Learning C is fun	✓

Passed all tests! ✓

Q4)

Input Format

You are given two strings, a and b, separated by a new line. Each string
lower-case Latin characters ('a'-'z').

Output Format

In the first line print two space-separated integers, representing the length
respectively.
In the second line print the string produced by concatenating a and b (a +
In the third line print two strings separated by a space, a' and b'. a' and b' a
a and b, respectively, except that their first characters are swapped.

Sample Input

abcd
ef

Sample Output

4 2
abcdef
ebcd af

Explanation

a = "abcd"
b = "ef"
|a| = 4
|b| = 2
a + b = "abcdef"
a' = "ebcd"
b' = "af"

Code:

```c
#include <stdio.h>
#include <string.h>

int main() {
    char a[101], b[101]; // Input strings (assuming maximum length of 100)

    // Read the input strings
    scanf("%s", a);
    scanf("%s", b);

    // Step 1: Print the lengths of a and b
    int len_a = strlen(a);
    int len_b = strlen(b);
    printf("%d %d\n", len_a, len_b);

    // Step 2: Print the concatenation of a and b
    printf("%s%s\n", a, b);

    // Step 3: Swap the first characters of a and b, then print the modified strings
    char temp = a[0];
    a[0] = b[0];
    b[0] = temp;
    printf("%s %s\n", a, b);

    return 0;
}
```

OUTPUT:

	Input	Expected	Got	
✓	abcd ef	4 2 abcdef ebcd af	4 2 abcdef ebcd af	✓

Passed all tests! ✓