“1684. Count the Number of Consistent Strings”

Question :

You are given a string allowed consisting of distinct characters and an array of strings words. A string is consistent if all characters in the string appear in the string allowed.

Return the number of consistent strings in the array words.

Constraints :

- $1 \leq$ words.length $\leq 10^4$
- $1 \leq$ allowed.length ≤ 26
- $1 \leq$ words[i].length ≤ 10
- The characters in allowed are distinct.
- words[i] and allowed contain only lowercase English letters.

Inputs :

- allowed: A string of distinct characters.
- words: An array of strings to check for consistency.

Outputs :

- An integer representing the count of consistent strings in words.

Example 1 :

Input:
allowed = "ab", words = ["ad","bd","aaab","baa","badab"]

Output:
2

Explanation:
Strings "aaab" and "baa" are consistent since they only contain characters 'a' and 'b'.

Example 2 :

Input:
allowed = "abc", words = ["a","b","c","ab","ac","bc","abc"]

Output:
7

Explanation:
All strings in words are consistent with allowed.

Example 3 :

Input:
allowed = "cad", words = ["cc","acd","b","ba","bac","bad","ac","d"]

Output:
4

Explanation:
Strings "cc", "acd", "ac", and "d" are consistent.

Algorithm :

1. Initialize a counter inconsistent to store the number of inconsistent strings.
2. Loop through each word in words:
    - Use strspn() to count the number of initial characters in the word that are present in allowed.
    - Compare the result of strspn() with the length of the word.
    - If they do not match, the word is inconsistent; otherwise, it is consistent.
3. Subtract the count of inconsistent strings from the total number of words to get the number of consistent strings.

4. Return the result.

Code :

```
#include <string.h>

int countConsistentStrings(char *allowed, char **words, int wordsSize) {
    int inconsistent = 0;

    for (int i = 0; i < wordsSize; i++) {
        if (strspn(words[i], allowed) != strlen(words[i])) {
            inconsistent++;
        }
    }
    return wordsSize - inconsistent;
}
```

Time Complexity :

- O(n * m):
    - n: Number of words in words.
    - m: Average length of each word.
    - Each call to strspn() checks up to mmm characters in the word.

Space Complexity :

- O(1): No additional space is used apart from a few variables.