

**PSG COLLEGE OF TECHNOLOGY, COIMBATORE -641004**  
**DEPARTMENT OF COMPUTER APPLICATIONS**  
**I SEMESTER MCA**  
**PROBLEM SHEET ON STRINGS**

- Given a string  $s$  consisting of words and spaces, return the length of the last word in the string. A word is a maximal substring consisting of non-space characters only.

Example 1:

Input:  $s = \text{"Hello World"}$

Output: 5

Explanation: The last word is "World" with length 5.

Example 2:

Input:  $s = \text{" fly me to the moon "}$

Output: 4

Explanation: The last word is "moon" with length 4.

Example 3:

Input:  $s = \text{"luffy is still joyboy"}$

Output: 6

Explanation: The last word is "joyboy" with length 6.

Constraints:

$1 \leq s.\text{length} \leq 104$

$s$  consists of only English letters and spaces ''.

There will be at least one word in  $s$ .

- A phrase is a palindrome if, after converting all uppercase letters into lowercase letters and removing all non-alphanumeric characters, it reads the same forward and backward.

Alphanumeric characters include letters and numbers.

Given a string  $s$ , return true if it is a palindrome, or false otherwise.

Example 1:

Input:  $s = \text{"A man, a plan, a canal: Panama"}$

Output: true

Explanation: "amanaplanacanalpanama" is a palindrome.

Example 2:

Input:  $s = \text{"race a car"}$

Output: false

Explanation: "raceacar" is not a palindrome.

Example 3:

Input:  $s = \text{" "}$

Output: true

Explanation: s is an empty string "" after removing non-alphanumeric characters.

Since an empty string reads the same forward and backward, it is a palindrome.

Constraints:

$1 \leq s.length \leq 2 * 10^5$

s consists only of printable ASCII characters.

3. Given an input string s, reverse the order of the words.

A word is defined as a sequence of non-space characters. The words in s will be separated by at least one space. Return a string of the words in reverse order concatenated by a single space.

Note that s may contain leading or trailing spaces or multiple spaces between two words. The returned string should only have a single space separating the words. Do not include any extra spaces.

Example 1:

Input: s = "the sky is blue"

Output: "blue is sky the"

Example 2:

Input: s = " hello world "

Output: "world hello"

Explanation: Your reversed string should not contain leading or trailing spaces. Example 3:

Input: s = "a good example"

Output: "example good a"

Explanation: You need to reduce multiple spaces between two words to a single space in the reversed string.

Constraints:

$1 \leq s.length \leq 10^4$

s contains English letters (upper-case and lower-case), digits, and spaces ' '.

. There is at least one word in s.

4. Given two strings s and t, determine if they are isomorphic.

Two strings s and t are isomorphic if the characters in s can be replaced to get t.

All occurrences of a character must be replaced with another character while preserving the order of characters. No two characters may map to the same character, but a character may map to itself.

Example 1:

Input: s = "egg", t = "add"

Output: true

Explanation:

The strings s and t can be made identical by:

Mapping 'e' to 'a'.

Mapping 'g' to 'd'.

Example 2:

Input: s = "foo", t = "bar"

Output: false

Explanation:

The strings s and t cannot be made identical as 'o' needs to be mapped to both 'a' and 'r'. Example 3:

Input: s = "paper", t = "title"

Output: true

Constraints:

$1 \leq s.length \leq 5 * 10^4$

$t.length == s.length$

s and t consist of any valid ascii character.

5. Given two strings s and t, return true if t is an anagram of s, and false otherwise. Note: An **anagram** is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

Example 1:

Input: s = "anagram", t = "nagaram"

Output: true

Example 2:

Input: s = "rat", t = "car"

Output: false

Constraints:

$1 \leq s.length, t.length \leq 5 * 10^4$

s and t consist of lowercase English letters.

6. Given a pattern and a string s, find if s follows the same pattern. Here follow means a full match, such that there is a bijection between a letter in pattern and a non-empty word in s. Specifically:

Each letter in pattern maps to exactly one unique word in s.

Each unique word in s maps to exactly one letter in pattern.

No two letters map to the same word, and no two words map to the same

letter. Example 1:

Input: pattern = "abba", s = "dog cat cat dog"

Output: true

Explanation:

The bijection can be established as:

'a' maps to "dog".

'b' maps to "cat".

Example 2:

Input: pattern = "abba", s = "dog cat cat fish"

Output: false

Example 3:

Input: pattern = "aaaa", s = "dog cat cat dog"

Output: false

Constraints:

$1 \leq \text{pattern.length} \leq 300$

pattern contains only lower-case English letters.

$1 \leq \text{s.length} \leq 3000$

s contains only lowercase English letters and spaces ''.

s does not contain any leading or trailing spaces.

All the words in s are separated by a single space.

7. Given a string array words, return the maximum value of  $\text{length}(\text{word}[i]) * \text{length}(\text{word}[j])$  where the two words do not share common letters. If no such two words exist, return 0.

Example 1:

Input: words = ["abcw", "baz", "foo", "bar", "xtfn", "abcdef"]

Output: 16

Explanation: The two words can be "abcw", "xtfn".

Example 2:

Input: words = ["a", "ab", "abc", "d", "cd", "bcd", "abcd"]

Output: 4

Explanation: The two words can be "ab", "cd".

Example 3:

Input: words = ["a", "aa", "aaa", "aaaa"]

Output: 0

Explanation: No such pair of words.

Constraints:

$2 \leq \text{words.length} \leq 1000$

$1 \leq \text{words[i].length} \leq 1000$

words[i] consists only of lowercase English letters.

8. Given a string s, reverse only all the vowels in the string and return it. The vowels are 'a', 'e', 'i', 'o', and 'u', and they can appear in both lower and upper cases, more than once.

Example 1:

Input: s = "IceCreAm"

Output: "AceCreIm"

**Explanation:**

The vowels in s are ['I', 'e', 'e', 'A']. On reversing the vowels, s becomes

"AceCreIm". Example 2:

Input: s = "leetcode"

Output: "leotcede"

**Constraints:**

$1 \leq s.length \leq 3 * 10^5$

s consist of printable ASCII characters.

9. Given an encoded string, return its decoded string.

The encoding rule is:  $k[\text{encoded\_string}]$ , where the `encoded_string` inside the square brackets is being repeated exactly  $k$  times. Note that  $k$  is guaranteed to be a positive integer. You may assume that the input string is always valid; there are no extra white spaces, square brackets are well-formed, etc. Furthermore, you may assume that the original data does not contain any digits and that digits are only for those repeat numbers,  $k$ . For example, there will not be input like `3a` or `2[4]`.

The test cases are generated so that the length of the output will never exceed 105.

**Example 1:**

Input: s = "3[a]2[bc]"

Output: "aaabcbc"

**Example 2:**

Input: s = "3[a2[c]]"

Output: "accaccacc"

**Example 3:**

Input: s = "2[abc]3[cd]ef"

Output: "abcabcccdcdcdef"

**Constraints:**

$1 \leq s.length \leq 30$

s consists of lowercase English letters, digits, and square brackets '[]'.

s is guaranteed to be a valid input.

All the integers in s are in the range [1, 300].

10. Given a string s and an integer k, return the length of the longest substring of s such that the frequency of each character in this substring is greater than or equal to k. If no such substring exists, return 0.

**Example 1:**

Input: s = "aaabb", k = 3

Output: 3

Explanation: The longest substring is "aaa", as 'a' is repeated 3 times.

**Example 2:**

Input: s = "ababbc", k = 2

Output: 5

Explanation: The longest substring is "ababb", as 'a' is repeated 2 times and 'b' is repeated 3 times.

**Constraints:**

$1 \leq s.length \leq 10^4$

s consists of only lowercase English letters.

$1 \leq k \leq 105$