PSG COLLEGE OF TECHNOLOGY, COIMBATORE -641004
DEPARTMENT OF COMPUTER APPLICATIONS
I SEMESTER MCA
23MX16 C PROGRAMMING LABORATORY
PROBLEM SHEET 10 – STRINGS

NOTE: Use pointers to access strings wherever possible

1. A wonderful string is a string where at most one letter appears an odd number of times.
- For example, "ccjjc" and "abab" are wonderful, but "ab" is not.

Given a string word that consists of the first ten lowercase English letters ('a' through 'j'), return the number of wonderful non-empty substrings in the word. If the same substring appears multiple times in word, then count each occurrence separately.
A substring is a contiguous sequence of characters in a string.

Example 1: Input: word = "aba" Output: 4 Explanation: The four wonderful substrings are underlined below: - "aba" -> "a" - "aba" -> "b" - "aba" -> "a" - "aba" -> "aba"	Example 2: Input: word = "aabb" Output: 9 Explanation: The nine wonderful substrings are underlined below: - "aabb" -> "a" - "aabb" -> "aa" - "aabb" -> "aab" - "aabb" -> "aabb" - "aabb" -> "a" - "aabb" -> "abb" - "aabb" -> "b" - "aabb" -> "bb" - "aabb" -> "b"
Example 3: Input: word = "he" Output: 2 Explanation: The two wonderful substrings are underlined below: - "he" -> "h" - "he" -> "e"	

2. You are given two strings s1 and s2 of equal length. A string swap is an operation where you choose two indices in a string (not necessarily different) and swap the characters at these indices. Return true if it is possible to make both strings equal by performing at most one string swap on exactly one of the strings. Otherwise, return false.

Example 1: Input: s1 = "bank", s2 = "kanb" Output: true Explanation: For example, swap the first character with the last character of s2 to make "bank".	**Example 2:** Input: s1 = "attack", s2 = "defend" Output: false Explanation: It is impossible to make them equal with one string swap.
Example 3: Input: s1 = "kelb", s2 = "kelb" Output: true	

Explanation: The two strings are already equal, so no string swap operation is required.	

3. The letter value of a letter is its position in the alphabet starting from 0 (i.e. 'a' -> 0, 'b' -> 1, 'c' -> 2, etc.).
The numerical value of some string of lowercase English letters s is the concatenation of the letter values of each letter in s, which is then converted into an integer.

- For example, if s = "acb", we concatenate each letter's letter value, resulting in "021". After converting it, we get 21.

You are given three strings firstWord, secondWord, and targetWord, each consisting of lowercase English letters 'a' through 'j' inclusive.
Return true if the summation of the numerical values of firstWord and secondWord equals the numerical value of targetWord, or false otherwise.

Example 1: Input: firstWord = "acb", secondWord = "cba", targetWord = "cdb" Output: true Explanation: The numerical value of firstWord is "acb" -> "021" -> 21. The numerical value of secondWord is "cba" -> "210" -> 210. The numerical value of targetWord is "cdb" -> "231" -> 231. We return true because 21 + 210 == 231.	**Example 2:** Input: firstWord = "aaa", secondWord = "a", targetWord = "aab" Output: false Explanation: The numerical value of firstWord is "aaa" -> "000" -> 0. The numerical value of secondWord is "a" -> "0" -> 0. The numerical value of targetWord is "aab" -> "001" -> 1. We return false because 0 + 0 != 1.
Example 3: Input: firstWord = "aaa", secondWord = "a", targetWord = "aaaa" Output: true Explanation: The numerical value of firstWord is "aaa" -> "000" -> 0. The numerical value of secondWord is "a" -> "0" -> 0. The numerical value of targetWord is "aaaa" -> "0000" -> 0. We return true because 0 + 0 == 0.	

4. A string s is called happy if it satisfies the following conditions:

- s only contains the letters 'a', 'b', and 'c'.
- s does not contain any of "aaa", "bbb", or "ccc" as a substring.
- s contains at most a occurrences of the letter 'a'.
- s contains at most b occurrences of the letter 'b'.
- s contains at most c occurrences of the letter 'c'.

Given three integers a, b, and c, return the longest possible happy string. If there are multiple longest happy strings, return any of them. If there is no such string, return the empty string "".
A substring is a contiguous sequence of characters within a string.

Example 1:	**Example 2:**
Input: a = 1, b = 1, c = 7 Output: "ccaccbcc" Explanation: "ccbccacc" would also be a correct answer.	Input: a = 7, b = 1, c = 0 Output: "aabaa" Explanation: It is the only correct answer in this case.

5. Given a string s, sort it in decreasing order based on the frequency of the characters. The frequency of a character is the number of times it appears in the string.
Return the sorted string. If there are multiple answers, return any of them.

Example 1:	**Example 2:**
Input: s = "tree" Output: "eert" Explanation: 'e' appears twice while 'r' and 't' both appear once. So 'e' must appear before both 'r' and 't'. Therefore "eetr" is also a valid answer.	Input: s = "cccaaa" Output: "aaaccc" Explanation: Both 'c' and 'a' appear three times, so both "cccaaa" and "aaaccc" are valid answers. Note that "cacaca" is incorrect, as the same characters must be together.
Example 3: Input: s = "Aabb" Output: "bbAa" Explanation: "bbaA" is also a valid answer, but "Aabb" is incorrect. Note that 'A' and 'a' are treated as two different characters.	

6. Given an input string (s) and a pattern (p), implement wildcard pattern matching with support for '?' and '*' where:

- '?' Matches any single character.
- '*' Matches any sequence of characters (including the empty sequence).

The matching should cover the entire input string (not partial).

Constraints:

- 0 <= s.length, p.length <= 2000

- s contains only lowercase English letters.
- p contains only lowercase English letters, '?' or '*'.

Example 1: Input: s = "aa", p = "a" Output: false Explanation: "a" does not match the entire string "aa".	**Example 2:** Input: s = "aa", p = "*" Output: true Explanation: '*' matches any sequence.
Example 3: Input: s = "cb", p = "?a" Output: false Explanation: '?' matches 'c', but the second letter is 'a', which does not match 'b'.	