**Date: 12.11.2024**

**Sri Vaigunth M-22AD134**

**DSA Coding Problems**

Set – 3

1. **Anagram:**

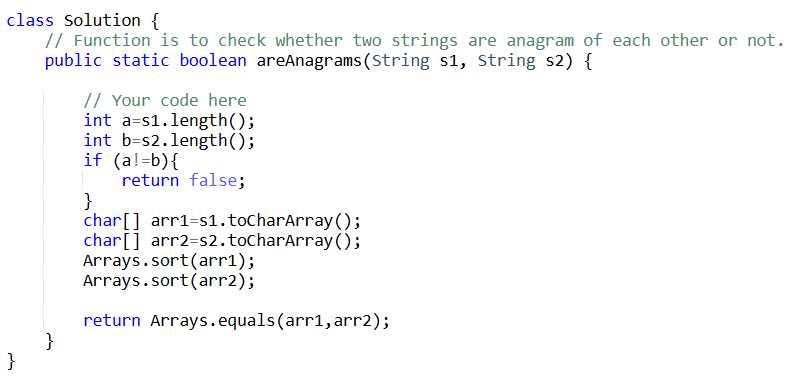
Given two strings **s1**and **s2**consisting of lowercase characters. The task is to check whether two given strings are an anagram of each other or not. An anagram of a string is another string that contains the same characters, only the order of characters can be different. For example, act and tac are an anagram of each other. Strings **s1**and **s2**can only contain lowercase alphabets.

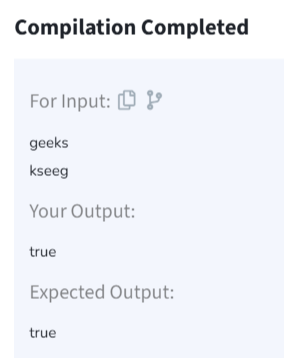
Note: You can assume both the strings s1 & s2 are **non-empty**.

**Input:** s1 = "geeks", s2 = "kseeg"

**Output:** true

**Code:**

****

****

**Time Complexity:** O(m\*log(m) + n\*log(n))

1. **Row with Max 1’s:**

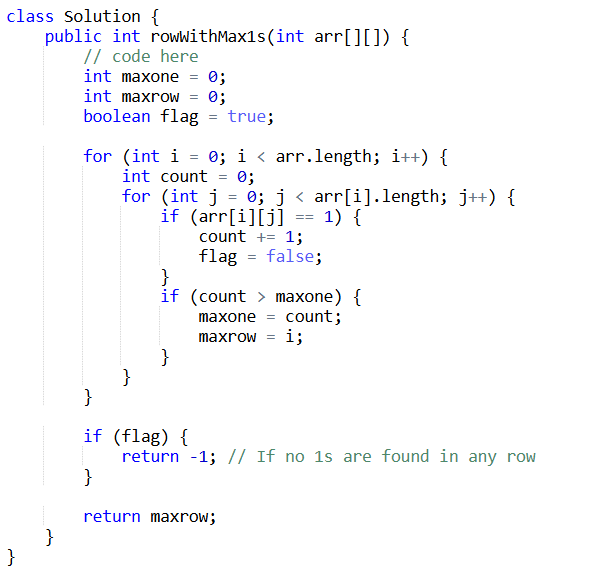
You are given a 2D array consistingof only **1's**and**0's**, where each row is sorted in non-decreasing order. You need to find and return the index of the first row that has the most number of 1s. If no such row exists, return **-1**.  
**Note:**0-based indexing is followed.

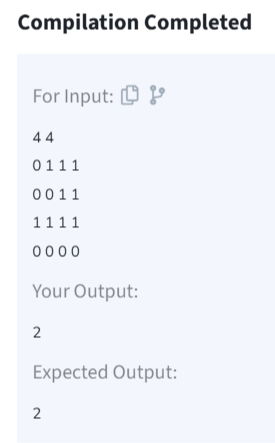
**Input:** arr[][] = [[0, 1, 1, 1], [0, 0, 1, 1], [1, 1, 1, 1], [0, 0, 0, 0]]

**Output:** 2

**Explanation:** Row 2 contains **4** 1's.

**Code:**





**Time Complexity: O(m\*n)**

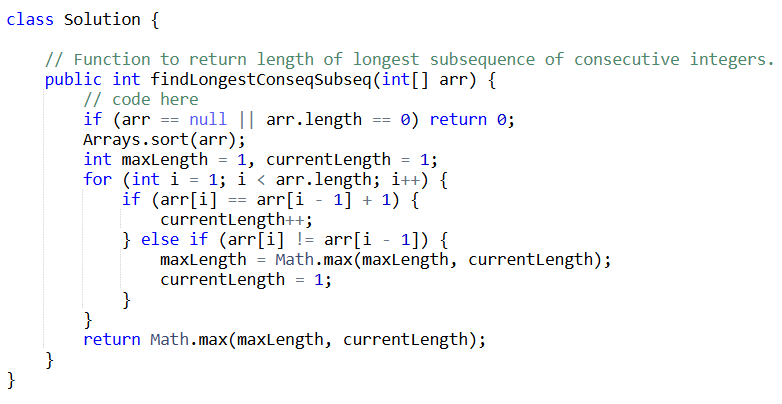
1. **Longest consecutive sequence:**

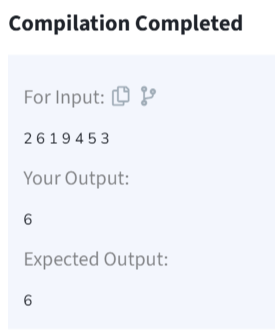
Given an array arr of non-negative integers. Find the length of the longest sub-sequence such that elements in the subsequence are consecutive integers, the consecutive numbers can be in any order.

**Input:** arr[] = [2, 6, 1, 9, 4, 5, 3]

**Output:** 6

**Code:**

****

****

**Time Complexity:** O(n log n)

1. **Longest palindrome in string**

Given a string **s**, your task is to find the longest palindromic substring within s. A **substring** is a contiguous sequence of characters within a string, defined as s[i...j] where 0 ≤ i ≤ j < len(s).

A **palindrome** is a string that reads the same forward and backward. More formally, s is a palindrome if reverse(s) == s.

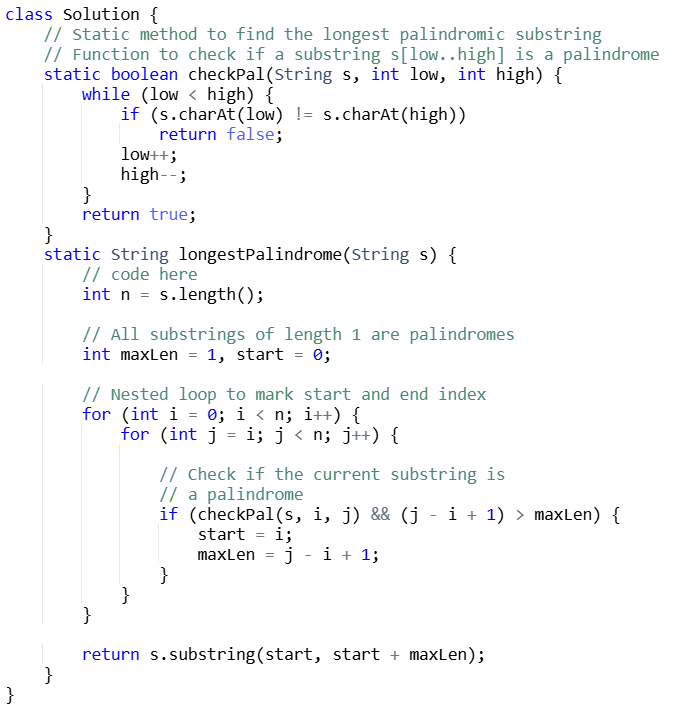
**Note:** If there are multiple palindromes with the same length, return the **first occurrence** of the longest palindromic substring from left to right.

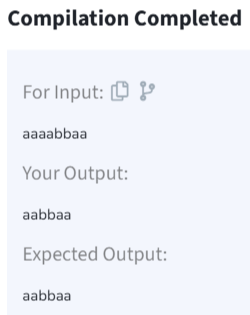
**Input:** s = "aaaabbaa"

**Output:** "aabbaa"

**Explanation**: The longest palindromic substring is "aabbaa".

**Code**:





**Time Complexity:** O(n^3)

1. **Rat in the maze**

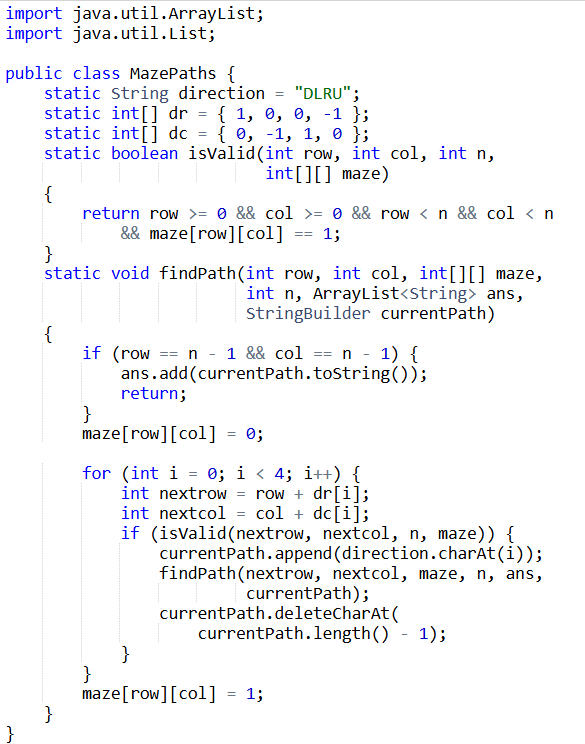
Consider a rat placed at **(0, 0)** in a square matrix **mat**of order **n\* n**. It has to reach the destination at **(n - 1, n - 1)**. Find all possible paths that the rat can take to reach from source to destination. The directions in which the rat can move are **'U'(up)**, **'D'(down)**, **'L' (left)**, **'R' (right)**. Value 0 at a cell in the matrix represents that it is blocked and rat cannot move to it while value 1 at a cell in the matrix represents that rat can be travel through it.  
**Note**: In a path, no cell can be visited more than one time. If the source cell is 0, the rat cannot move to any other cell. In case of no path, return an empty list. The driver will output **"-1"** automatically.

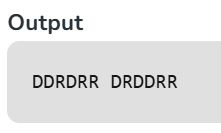
**Input**: mat[][] = [[1, 0, 0, 0], [1, 1, 0, 1], [1, 1, 0, 0], [0, 1, 1, 1]]

**Output:** DDRDRR DRDDRR

**Explanation**: The rat can reach the destination at (3, 3) from (0, 0) by two paths - DRDDRR and DDRDRR, when printed in sorted order we get DDRDRR DRDDRR.

**Code**:





**Time Complexity:** O(3^(m\*n))