****

**Technical Assessment**

**Question no 1:**

You are given a string **s** of lower case english alphabets. You can choose any two characters in the string and replace all the occurences of the first character with the second character and replace all the occurences of the second character with the first character. Your aim is to find the lexicographically smallest string that can be obtained by doing this operation at most once.

**Example 1:**

**Input**:

A = "ccad"

**Output:** "aacd"

**Explanation**:

In ccad, we choose ‘a’ and ‘c’ and after

doing the replacement operation once we get,

aacd and this is the lexicographically

smallest string possible.

**Example 2:**

**Input:**

A = "abba"

**Output:** "abba"

**Explanation:**

In abba, we can get baab after doing the

replacement operation once for ‘a’ and ‘b’

but that is not lexicographically smaller

than abba. So, the answer is abba.

**Your Task:** You don't need to read input or print anything. Your task is to complete the function **chooseandswap()** which takes the string A as input parameters and returns the **lexicographically smallest string** that is possible after doing the operation at most once.  **Expected Time Complexity:** O(|A|) length of the string A **Expected Auxiliary Space:** O(1)

**Constraints:** 1<= |A| <=105

**Question: 2**

Given a list of non negative integers, arrange them in such a manner that they form the largest number possible.The result is going to be very large, hence return the result in the form of a string.

**Example 1:**

**Input:**

N = 5

Arr[] = {3, 30, 34, 5, 9}

**Output:** 9534330

**Explanation:** Given numbers are {3, 30, 34,

5, 9}, the arrangement 9534330 gives the

largest value.

**Example 2:**

**Input:**

N = 4

Arr[] = {54, 546, 548, 60}

**Output:** 6054854654

**Explanation:** Given numbers are {54, 546,

548, 60}, the arrangement 6054854654

gives the largest value.

**Your Task:** You don't need to read input or print anything. Your task is to complete the function **printLargest()** which takes the **array of strings** **arr[]**as parameter and returns a string denoting the answer.  **Expected Time Complexity:** O(NlogN) **Expected Auxiliary Space:** O(1)

**Constraints:** 1 ≤ N ≤ 105 0 ≤ Arr[i] ≤ 1018

**Question no 3**

Given a string S, find the longest palindromic substring in S. Substring of string S: S[ i . . . . j ] where 0 ≤ i ≤ j < len(S). Palindrome string: A string which reads the same backwards. More formally, S is palindrome if reverse(S) = S. Incase of conflict, return the substring which occurs first ( with the least starting index).

Example 1:

Input:

S = "aaaabbaa"

Output: aabbaa

Explanation: The longest Palindromic

substring is "aabbaa".

â€‹Example 2:

Input:

S = "abc"

Output: a

Explanation: "a", "b" and "c" are the

longest palindromes with same length.

The result is the one with the least

starting index.

 Your Task: You don't need to read input or print anything. Your task is to complete the function longestPalin() which takes the string S as input and returns the longest palindromic substring of S.

 Expected Time Complexity: O(|S|). Expected Auxiliary Space: O(1).

 Constraints: 1<=|S|<=103

**Question no 4**

There is **one** meeting room in a firm. There are **N** meetings in the form of **(S[i], F[i])** where **S[i]**is start time of meeting **i**and **F[i]**is finish time of meeting **i.** What is the **maximum** number of meetings that can be accommodated in the meeting room when only one meeting can be held in the meeting room at a particular time? Also note start time of one chosen meeting can't be equal to the end time of the other chosen meeting.

**Example 1:**

**Input:**

N = 6

S[] = {1,3,0,5,8,5}

F[] = {2,4,6,7,9,9}

**Output:**

4

**Explanation:**

Four meetings can be held with

given start and end timings.

**Example 2:**

**Input:**

N = 8

S[] = {75250, 50074, 43659, 8931, 11273,

27545, 50879, 77924}

F[] = {112960, 114515, 81825, 93424, 54316,

35533, 73383, 160252}

**Output:**

3

**Explanation:**

Only three meetings can be held

with given start and end timings.

**Your Task** : You don't need to read inputs or print anything. Complete the function **maxMeetings()**that recieves array **S[ ]**and **F[ ]**along with their size **N** as input parameters and returns the maximum number of meetings that can be held in the meeting room.

**Expected Time Complexity**: O(N\*LogN) **Expected Auxilliary Space** : O(N)

**Constraints:** 1 ≤ N ≤ 105 0 ≤ S[i] < F[i] ≤ 105