**📝 Infinite Champions Programme – Day 4 (Assignment Sheet)**

**📌 Instructions  
• Deadline: Submit your solutions by 30th September, 2025, EOD.  
• Platform: Test your solutions on LeetCode  
• Collaboration: Discussing concepts is encouraged, but all code must be your own.**

1. [**Longest Substring Without Repeating Characters (3)**](https://leetcode.com/problems/longest-substring-without-repeating-characters/)  
   • Problem: Given a string s, find the length of the longest substring without repeating characters.  
   • Objective: Use the sliding window technique with a hash set/map to efficiently track unique characters.  
   • YouTube Solution (Java): [Longest Substring Without Repeating Characters – Java Solution](https://www.youtube.com/results?search_query=Longest+Substring+Without+Repeating+Characters+LeetCode+Java)

class Solution {

    public int lengthOfLongestSubstring(String s) {

        int ans=0;

        int i=0;

        int j=0;

        int[] v=new int[128];

        while(j<s.length()){

            v[s.charAt(j)]++;

            while(v[s.charAt(j)]>1){

                v[s.charAt(i)]--;

                i++;

            }

            ++j;

            ans=Math.max(ans,j-i);

        }

        return ans;

    }

}

1. [**Permutation in String (567)**](https://leetcode.com/problems/permutation-in-string/)  
   • Problem: Given two strings s1 and s2, return true if s2 contains a permutation of s1, otherwise false.  
   • Objective: Use sliding window + character frequency counting to check for permutations.  
   • YouTube Solution (Java): [Permutation in String – Java Solution](https://www.youtube.com/results?search_query=Permutation+in+String+LeetCode+Java)
2. [**Find All Anagrams in a String (438)**](https://leetcode.com/problems/find-all-anagrams-in-a-string/)  
   • Problem: Given two strings s and p, return all start indices of p’s anagrams in s.  
   • Objective: Apply sliding window + frequency comparison to find anagrams efficiently.  
   • YouTube Solution (Java): [Find All Anagrams in a String – Java Solution](https://www.youtube.com/results?search_query=Find+All+Anagrams+in+a+String+LeetCode+Java)

class Solution {

    public List<Integer> findAnagrams(String s, String p) {

        List<Integer> result = new ArrayList<>();

        if (s.length() < p.length()){

            return result;

        }

        int[] freq = new int[26];

        for (char c : p.toCharArray()) {

            freq[c - 'a']++;

        }

        int i = 0;

        int j = 0;

        int needed = p.length();

        while (j < s.length()) {

            if (freq[s.charAt(j) - 'a'] > 0) {

                needed--;

            }

            freq[s.charAt(j) - 'a']--;

            j++;

            if (j - i == p.length()) {

                if (needed == 0) {

                    result.add(i);

                }

                if (freq[s.charAt(i) - 'a'] >= 0) {

                    needed++;

                }

                freq[s.charAt(i) - 'a']++;

                i++;

            }

        }

    return result;

    }

}

1. [**Longest Repeating Character Replacement (424)**](https://leetcode.com/problems/longest-repeating-character-replacement/)  
   • Problem: Given a string s and an integer k, return the length of the longest substring where you can replace at most k characters to make all characters identical.  
   • Objective: Use sliding window + character frequency count to dynamically expand/shrink the window.  
   • YouTube Solution (Java): [Longest Repeating Character Replacement – Java Solution](https://www.youtube.com/results?search_query=Longest+Repeating+Character+Replacement+LeetCode+Java)

class Solution {

    public int characterReplacement(String s, int k) {

        int[] freq = new int[26];

        int i = 0, max1 = 0, ans = 0, j = 0;

        while (j < s.length()) {

            freq[s.charAt(j) - 'A']++;

            max1 = Math.max(max1, freq[s.charAt(j) - 'A']);

            while ((j - i + 1) - max1 > k) {

                freq[s.charAt(i) - 'A']--;

                i++;

            }

            ans = Math.max(ans, j - i + 1);

            j++;

        }

        return ans;

    }

}

1. [**Max Consecutive Ones III (1004)**](https://leetcode.com/problems/max-consecutive-ones-iii/)  
   • Problem: Given a binary array nums and an integer k, return the maximum number of consecutive 1s in the array if you can flip at most k zeros.  
   • Objective: Use sliding window to count and control the number of zeros inside the window.  
   • YouTube Solution (Java): [Max Consecutive Ones III – Java Solution](https://www.youtube.com/results?search_query=Max+Consecutive+Ones+III+LeetCode+Java)

class Solution {

    public int longestOnes(int[] nums, int k) {

        int i= 0;

        int j=0;

        int zeros = 0;

        int ans = 0;

        while (j< nums.length) {

            if (nums[j] == 0) zeros++;

            while (zeros > k) {

                if (nums[i] == 0) zeros--;

                i++;

            }

            j++;

            ans = Math.max(ans, j-i);

        }

        return ans;

    }

}

**📚 Submission Checklist  
• Time and space complexity analysis for each solution.  
• Test cases demonstrating the correctness of your solutions.**