



CSE3032 - Competitive Programming
WIN SEM (2022-2023) AMR
Class Number: AP2022236001007
Slot: L11+L12+L19+L20
ASSIGNMENT - 10

Last Date for Submission: Thursday (20-04-2023) @ 12.40PM

Name: Chilamakuru Sai Venkata Yaswanth

Reg No: 20BCI7076

Write the program using (C / C++ / Java / Python) to solve the following problems.

Concept: Sliding Window Technique

S.No	Problem Name	Statement
1	Longest Substring Without Repeating Characters	<p>Given a string s, find the length of the longest substring without repeating characters.</p> <p>Input: s = "abcabcbb" Output: 3 Explanation: The answer is "abc", with the length of 3.</p> <p>Input: s = "bbbbbb" Output: 1 Explanation: The answer is "b", with the length of 1.</p> <p>Input: s = "pwwkew" Output: 3 Explanation: The answer is "wke", with the length of 3. Notice that the answer must be a substring, "pwke" is a subsequence and not a substring.</p>

2	Minimum Size Subarray Sum	<p>Given an array of positive integers nums and a positive integer target, return the minimal length of a Subarray whose sum is greater than or equal to target. If there is no such subarray, return 0 instead.</p> <p>Input: target = 7, nums = [2,3,1,2,4,3] Output: 2 Explanation: The subarray [4,3] has the minimal length under the problem constraint.</p> <p>Input: target = 4, nums = [1,4,4] Output: 1</p> <p>Input: target = 11, nums = [1,1,1,1,1,1,1] Output: 0</p> <p>Follow up: If you have figured out the $O(n)$ solution, try coding another solution of which the time complexity is $O(n \log(n))$.</p>
3	Find All Anagrams in a String	<p>Given two strings s and p, return an array of all the start indices of p's anagrams in s. You may return the answer in any order.</p>

		<p>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.</p> <p>Input: s = "cbaebabacd", p = "abc" Output: [0,6] Explanation: The substring with start index = 0 is "cba", which is an anagram of "abc". The substring with start index = 6 is "bac", which is an anagram of "abc".</p> <p>Input: s = "abab", p = "ab" Output: [0,1,2] Explanation: The substring with start index = 0 is "ab", which is an anagram of "ab". The substring with start index = 1 is "ba", which is an anagram of "ab". The substring with start index = 2 is "ab", which is an anagram of "ab".</p>
--	--	--

4	Minimum Window Substring	<p>Given two strings s and t of lengths m and n respectively, return the minimum window substring of s such that every character in t (including duplicates) is included in the window. If there is no such substring, return the empty string "".</p> <p>The testcases will be generated such that the answer is unique.</p> <p>Input: s = "ADOBECODEBANC", t = "ABC" Output: "BANC" Explanation: The minimum window substring "BANC" includes 'A', 'B', and 'C' from string t.</p> <p>Input: s = "a", t = "a" Output: "a" Explanation: The entire string s is the minimum window.</p> <p>Input: s = "a", t = "aa" Output: "" Explanation: Both 'a's from t must be included in the window. Since the largest window of s only has one 'a', return empty string.</p>
5	Permutation in String	<p>Given two strings s1 and s2, return true if s2 contains a permutation of s1, or false otherwise.</p> <p>In other words, return true if one of s1's permutations is the substring of s2.</p> <p>Input: s1 = "ab", s2 = "eidbaooo" Output: true Explanation: s2 contains one permutation of s1 ("ba").</p> <p>Input: s1 = "ab", s2 = "eidboao" Output: false</p>
6	Minimum Window Subsequence	<p>You are given two strings 'S' and 'T'. Your task is to find the minimum (Contiguous) substring 'W' of 'S', such that 'T' is a subsequence of 'W'</p> <p>A subsequence is a sequence that can be derived from another sequence by removing zero or more elements, without changing the order.</p> <p>A substring is a contiguous part of a string.</p> <p>Input: S = "abcdebddde", T = "bde" Output: "bcde" Explanation: There are many substrings with "bde" but the smallest amongst them is "bcde" and "bdde" of length 4. Out of these 2, "bcde" occurs first, Hence it is the answer.</p>

7	Subarrays with K Different Integers	<p>Given an integer array nums and an integer k, return the number of good subarrays of nums.</p> <p>A good array is an array where the number of different integers in that array is exactly k.</p> <p>For example, [1,2,3,1,2] has 3 different integers: 1, 2, and 3. A subarray is a contiguous part of an array.</p> <p>Input: nums = [1,2,1,2,3], k = 2 Output: 7 Explanation: Subarrays formed with exactly 2 different integers: [1,2], [2,1], [1,2], [2,3], [1,2,1], [2,1,2], [1,2,1,2]</p> <p>Input: nums = [1,2,1,3,4], k = 3 Output: 3 Explanation: Subarrays formed with exactly 3 different integers: [1,2,1,3], [2,1,3], [1,3,4].</p>
8	Sliding Window Maximum (Maximum of all subarrays of size k).	<p>Given an array arr of size N and an integer K, the task is to find the maximum for each and every contiguous subarray of size K.</p> <p>Input: arr[] = {1, 2, 3, 1, 4, 5, 2, 3, 6}, K = 3 Output: 3 3 4 5 5 5 6 All contiguous subarrays of size k are {1, 2, 3} => 3 {2, 3, 1} => 3 {3, 1, 4} => 4 {1, 4, 5} => 5 {4, 5, 2} => 5 {5, 2, 3} => 5 {2, 3, 6} => 6</p> <p>Input: arr[] = {8, 5, 10, 7, 9, 4, 15, 12, 90, 13}, K = 4 Output: 10 10 10 15 15 90 90</p>

Note:

- If Code similarity is found, assignment will not be considered and Zero (0) Marks will be awarded.
- You have to upload a single document consisting of all the above programs and corresponding Output.

Reference:

- <https://www.geeksforgeeks.org/window-sliding-technique/>
- <https://itnext.io/sliding-window-algorithm-technique-6001d5f8b3>
- <https://takeuforward.org/data-structure/sliding-window-technique/>

1) Longest substring without Repeating characters:

Code:

```
import java.util.Scanner;
public class Assignments {
    public static int longest(String s) {
        int n=s.length();
        int a=0;
        int start=0;
        int end=0;
        int[] freq=new int[128];
        while(end<n) {
            char c=s.charAt(end);
            freq[c]++;
            while(freq[c]>1) {
                char d=s.charAt(start);
                freq[d]--;
                start++;
            }
            end++;
            a=Math.max(a,end-start);
        }
        return a;
    }

    public static void main(String [] args) {
        Scanner sc=new Scanner(System.in);
        System.out.print("");
        String s=sc.next();
        int length=longest(s);
        System.out.println("" + length);
    }
}
```

Output:

```
C:\Users\Windows\OneDrive\Documents\CP>javac Assignments.java

C:\Users\Windows\OneDrive\Documents\CP>java Assignments.java
abcabcbb
3

C:\Users\Windows\OneDrive\Documents\CP>java Assignments.java
bbbbbb
1

C:\Users\Windows\OneDrive\Documents\CP>java Assignments.java
pwwkew
3

C:\Users\Windows\OneDrive\Documents\CP>|
```

2)Minimum size subarray sum:

Code:

```
import java.util.*;

public class Assignments{
    public static boolean found=false;
    public static int ans=0;
    public static int slide(int arr[], int n, int k) {
        int currsum = 0;
        for (int i = 0; i < k; i++) {
            currsum += arr[i];
        }
        if(currsum>=n) {
            found=true;
            return k;
        }
        for (int i = k; i < arr.length ; i++) {
            currsum -= arr[i - k];
            currsum += arr[i];
            if(currsum>=n) {
                found=true;
                ans=k;
                return k;
            }
        }
    }
}
```

```

    }
    }
    return 0;
}
public static void main(String args[]) {
    Scanner sc=new Scanner(System.in);
    int n=sc.nextInt();
    int arr[]=new int[n];
    for(int i=0;i<n;i++) {
        arr[i]=sc.nextInt();
    }
    int k=sc.nextInt();
    int i=1;
    while(!found&& i<=arr.length) {
        slide(arr,k,i);

        i++;
    }
    System.out.println(ans);
}
}

```

Output:

```

C:\Users\Windows\OneDrive\Documents\CP>javac Assignments.java

C:\Users\Windows\OneDrive\Documents\CP>java Assignments.java
6
2 3 1 2 4 3
7
2
C:\Users\Windows\OneDrive\Documents\CP>

```

3) Find all Anagrams in a String:

Code:

```
import java.util.*;
public class Assignments {
    public static void main(String [] args) {
        Scanner sc=new Scanner(System.in);
        System.out.print("s:");
        String s=sc.nextLine();
        System.out.print("p:");
        String p=sc.nextLine();
        List<Integer> r=anagram(s,p);
        System.out.println(r);
    }
    public static List<Integer> anagram(String s,String p){
        List<Integer> r=new ArrayList<>();
        if(s==null || s.length()==0 || p==null || p.length()==0 ||
s.length()<p.length()){
            return r;
        }
        int[] count=new int[26];
        for(char c:p.toCharArray()){
            count[c-'a']++;
        }
        int left=0;
        int right=0;
        int len=p.length();
        while(right<s.length()){
            if(count[s.charAt(right)-'a']>0){
                count[s.charAt(right)-'a']--;
                right++;
                len--;
            }
            else{
                count[s.charAt(left)-'a']++;
                left++;
                len++;
            }
        }
        if(len==0){
```



```

        r.add(left);
    }
}
return r;
}

```

Output:

```

C:\Users\Windows\OneDrive\Documents\CP>javac Assignments.java

C:\Users\Windows\OneDrive\Documents\CP>java Assignments.java
s:cbaebabacd
p:abc
[0, 6]

C:\Users\Windows\OneDrive\Documents\CP>java Assignments.java
s:abab
p:ab
[0, 1, 2]

C:\Users\Windows\OneDrive\Documents\CP>|

```

4) Minimum window substring:

Code:

```

import java.util.*;

public class Assignments {
    static boolean found=false;

    public static boolean checker(String str,String c) {
        for(int i=0;i<c.length();i++) {
            String ch=c.charAt(i)+"";
            if(!str.contains(ch)) {
                return false;
            }
        }

        return true;
    }

    public static void slide(String s,String c,int k) {
        String str=s.substring(0,k);
        String ans="";
        if(checker(str,c)) {
            //System.out.println(str);

```

```

        ans+=str;
        found=true;
    }
    for(int i=k;i<s.length();i++) {
        if(!found) {
            str=str.substring(1)+s.charAt(i);
            if(checker(str,c)) {
                //System.out.println(str);
                ans+=str;
                found=true;
            }
        }
    }

    System.out.println(ans);
}

public static void main(String args[]) {
    Scanner sc=new Scanner(System.in);
    String s=sc.nextLine();
    String c=sc.nextLine();
    int i=c.length();
    while(!found&& i<=s.length()) {
        slide(s,c,i);
        i++;
    }
}
}

```

Output:

```

C:\Users\Windows\OneDrive\Documents\CP>javac Assignments.java

C:\Users\Windows\OneDrive\Documents\CP>java Assignments.java
ADOBECODEBANC
ABC

BANC

C:\Users\Windows\OneDrive\Documents\CP>|

```

5) Permutation in String:

Code:

```
import java.util.*;
public class Assignments{
    static boolean con=false;
    public static void per(String str,String ans,String c){
        if (str.length()==0){
            if(c.contains(ans)){
                con=true;
            }
            return;
        }
        for (int i=0;i<str.length();i++){
            char ch=str.charAt(i);
            String ros=str.substring(0,i)+str.substring(i+1);
            per(ros,ans+ch,c);
        }
    }
    public static void main(String [] args){
        Scanner sc=new Scanner(System.in);
        String s=sc.nextLine();
        String c=sc.nextLine();
        per(s,"",c);
        System.out.println(con);
    }
}
```

Output:

```
C:\Users\Windows\OneDrive\Documents\CP>javac Assignments.java
C:\Users\Windows\OneDrive\Documents\CP>java Assignments.java
ab
eidbaooo
true
C:\Users\Windows\OneDrive\Documents\CP>|
```

6) Minimum window subsequence:

Code:

```
import java.util.*;
public class Assignments {
    static boolean found=false;
    public static boolean checker(String str,String c) {
        ArrayList<String> list=new ArrayList<>();
        String ans="";
        findsubsequences(str,ans,list);
        if(list.contains(c)) {
            return true;
        }
        return false;
    }
    private static void findsubsequences(String s,String
ans,ArrayList<String> list){
        if (s.length() == 0) {
            list.add(ans);
            return;
        }
        findsubsequences(s.substring(1), ans + s.charAt(0), list);
        findsubsequences(s.substring(1), ans,list);
    }
    public static void slide(String s,String c,int k) {
        String str=s.substring(0,k);
        if(checker(str,c)) {
            System.out.println(str);
            found=true;
        }
        for(int i=k;i<s.length();i++) {
            if(!found) {
                str=str.substring(1)+s.charAt(i);
                if(checker(str,c)) {
                    System.out.println(str);
                    found=true;
                }
            }
        }
    }
}
```

```

    }
    public static void main(String args[]) {
        Scanner sc=new Scanner(System.in);
        String s=sc.nextLine();
        String c=sc.nextLine();
        int i=c.length();
        while(!found&& i<=s.length()) {
            slide(s,c,i);
            i++;
        }
    }
}

```

Output:

```

C:\Users\Windows\OneDrive\Documents\CP>javac Assignments.java

C:\Users\Windows\OneDrive\Documents\CP>java Assignments.java
abcdebdde
bde
bcde

C:\Users\Windows\OneDrive\Documents\CP>|

```

7) Subarrays with K different Integers:

Code:

```

import java.util.*;
public class Assignments{
    public static int count=0;
    public static void slide(int arr[],int k,int check) {
        ArrayList<Integer> list= new ArrayList<>();
        for(int i=0;i<k;i++) {
            list.add(arr[i]);
        }
        if(good_arr(list,check)) count++;
        for(int i=k;i<arr.length;i++) {
            list.remove(0);
            list.add(arr[i]);
            if(good_arr(list,check)) count++;
        }
    }
}

```

```

    }
}
public static boolean good_arr(ArrayList<Integer> list,int k) {
    ArrayList<Integer> list2= new ArrayList<>();
    for(int i=0;i<list.size();i++) {
        if(!list2.contains(list.get(i))) {
            list2.add(list.get(i));
        }
    }
    if(list2.size()==k) {
        return true;
    }else {
        return false;
    }
}
public static void main(String [] args) {
    Scanner sc=new Scanner(System.in);
    int n=sc.nextInt();
    int arr[]=new int[n];
    for(int i=0;i<n;i++) {
        arr[i]=sc.nextInt();
    }
    int k=sc.nextInt();
    for(int i=n;i>=k;i--) {
        slide(arr,i,k);
    }
    System.out.println(count);
}
}

```

Output:

```

C:\Users\Windows\OneDrive\Documents\CP>javac Assignments.java

C:\Users\Windows\OneDrive\Documents\CP>java Assignments.java
5
1 2 1 2 3
2
7

C:\Users\Windows\OneDrive\Documents\CP>

```

8) Sliding Window Maximum (Maximum of all subarrays of size k).

Code:

```
import java.util.*;
public class Assignments {
    public static void maxofqueue(Queue<Integer> q) {
        int max=0;
        for(int i=0;i<q.size();i++) {
            int m=q.peek();
            if(m>max) {
                max=m;
            }
            q.remove();
            q.add(m);
        }
        System.out.print(max+" ");
    }
    public static void main(String args[]) {
        Scanner sc=new Scanner(System.in);
        int n=sc.nextInt();
        int arr[]=new int[n];
        for(int i=0;i<n;i++) {
            arr[i]=sc.nextInt();
        }
        int k=sc.nextInt();
        Queue<Integer> q=new LinkedList<>();
        for(int i=0;i<k;i++) {
            q.add(arr[i]);
        }
        for(int i=k;i<n;i++) {
            maxofqueue(q);
            q.remove();
            q.add(arr[i]);
        }
        maxofqueue(q);
    }
}
```

Output:

```
C:\Users\Windows\OneDrive\Documents\CP>javac Assignments.java  
C:\Users\Windows\OneDrive\Documents\CP>java Assignments.java  
9  
1 2 3 1 4 5 2 3 6  
3  
3 3 4 5 5 5 6  
C:\Users\Windows\OneDrive\Documents\CP>
```