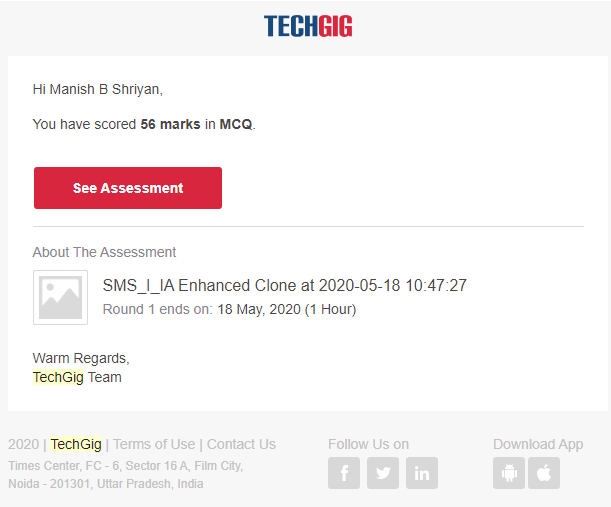
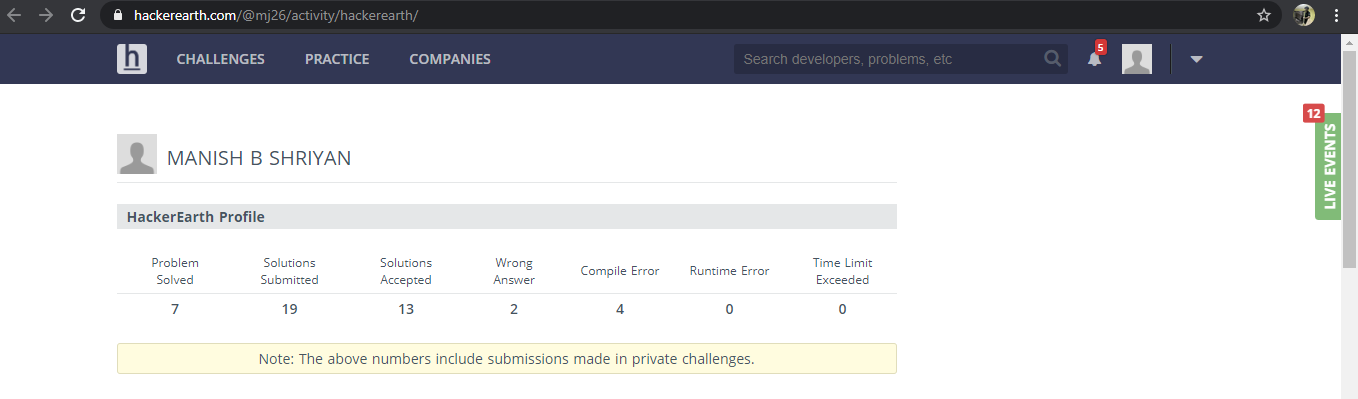
**DAILY ONLINE ACTIVITIES SUMMARY**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date:** | **18/05/2020** | | | | | **Name:** | **Manish B Shriyan** | |
| **Sem & Sec** | **8th sem B sec** | | | | | **USN:** | **4AL16CS131** | |
| **Online Test Summary** | | | | | | | | |
| **Subject** | | **SMS** | | | | | | |
| **Max. Marks** | | **60** | | **Score** | | | **56** | |
| **Certification Course Summary** | | | | | | | | |
| **Course** | **HackerEarth Challenges Solved** | | | | | | | |
| **Certificate Provider** | | | **HackerEarth** | | **Duration** | | | **5 hours** |
| **Coding Challenges** | | | | | | | | |
| **Problem Statement:**  **Write down a java program to print even and odd numbers series respectively from two threads: t1 and t2 synchronizing on a shared object Let t1 print message “ping — >” and t2 print message “,—-pong”. Take as command line arguments, the following inputs to the program: Sleep Interval for thread t1 Write down a java program to print even and odd numbers series respectively from two threads: t1 and t2 synchronizing on a shared object** | | | | | | | | |
| **Status: Solved** | | | | | | | | |
| **Uploaded the report in Github** | | | | | **Uploaded** | | | |
| **If yes Repository name** | | | | | **ManishShriyan** | | | |
| **Uploaded the report in slack** | | | | | **Yes** | | | |

Online Test Details:



Certification Course Details:



Coding Challenges Details:

|  |
| --- |
| public class PingPong extends Thread { |
|  | static StringBuilder object = new StringBuilder(""); | |
|  |  | |
|  | public static void main(String[] args) throws InterruptedException { | |
|  |  | |
|  | Thread t1 = new PingPong(); | |
|  | Thread t2 = new PingPong(); | |
|  |  | |
|  | t1.setName("\nping"); | |
|  | t2.setName(" pong"); | |
|  |  | |
|  | t1.start(); | |
|  | t2.start(); | |
|  | } | |
|  |  | |
|  | @override | |
|  | public void run() { | |
|  | working(); | |
|  | } | |
|  |  | |
|  | void working() { | |
|  | while (true) { | |
|  | synchronized (object) { | |
|  | try { | |
|  | System.out.print(Thread.currentThread().getName()); | |
|  | object.notify(); | |
|  | object.wait(); | |
|  | } catch (InterruptedException e) { | |
|  | e.printStackTrace(); | |
|  | } | |
|  | } | |
|  | } | |
|  | } | |
|  | } | |
|  |  | |
|  |  | |
|  |  | |
|  | ======================================================================================================================== | |
|  | ======================================================================================================================== | |
|  | package pk; | |
|  | import java.util.Scanner; | |
|  | public class StringOperators | |
|  | { | |
|  | public static void main(String args[]) | |
|  | { | |
|  | int i; | |
|  | String str; | |
|  |  | |
|  | int counter[] = new int[256]; | |
|  | Scanner in = new Scanner(System.in); | |
|  |  | |
|  | System.out.print("Enter a String : "); | |
|  | str=in.nextLine(); | |
|  |  | |
|  | for (i = 0; i < str.length(); i++) { | |
|  | counter[(int) str.charAt(i)]++; | |
|  | } | |
|  | // Print Frequency of characters | |
|  | for (i = 0; i < 256; i++) { | |
|  | if (counter[i] != 0) { | |
|  | System.out.println((char) i + ":-" + counter[i] + " times"); | |
|  | } | |
|  | } | |
|  | } | |
|  | } | |
| package shortestpalindromeexample.java; | |
|  | | import java.util.Scanner; |
|  | |  |
|  | | public class ShortestPalindromeDemo { |
|  | |  |
|  | | public static String shortestPalindrome(String str) { |
|  | |  |
|  | | int x=0; |
|  | | int y=str.length()-1; |
|  | |  |
|  | | while(y>=0){ |
|  | | if(str.charAt(x)==str.charAt(y)){ |
|  | | x++; |
|  | | } |
|  | | y--; |
|  | | } |
|  | |  |
|  | | if(x==str.length()) |
|  | | return str; |
|  | |  |
|  | | String suffix = str.substring(x); |
|  | | String prefix = new StringBuilder(suffix).reverse().toString(); |
|  | | String mid = shortestPalindrome(str.substring(0, x)); |
|  | |  |
|  | | return prefix+mid+suffix; |
|  | | } |
|  | |  |
|  | | public static void main(String[] args) { |
|  | |  |
|  | | Scanner in = new Scanner(System.in); |
|  | |  |
|  | | System.out.println("Enter a String to find out shortest palindrome"); |
|  | |  |
|  | | String str=in.nextLine(); |
|  | |  |
|  | | System.out.println("Shortest palindrome of "+str+" is "+shortestPalindrome(str)); |
|  | |  |
|  | | } |
|  | |  |
|  | |  |
|  | | ======================================================================================================================================== |
|  | | ======================================================================================================================================== |
|  | |  |
|  | |  |
|  | | import java.util.Stack; |
|  | |  |
|  | | // Data Structure to store a linked list node |
|  | | class Node { |
|  | | int data; |
|  | | Node next; |
|  | |  |
|  | | Node(int i) |
|  | | { |
|  | | this.data = i; |
|  | | this.next = null; |
|  | | } |
|  | | }; |
|  | |  |
|  | | class Main |
|  | | { |
|  | | // Function to determine if a given linked list is palindrome or not |
|  | | public static boolean isPalindrome(Node head) |
|  | | { |
|  | | // construct an empty stack |
|  | | Stack<Integer> s = new Stack<>(); |
|  | |  |
|  | | // push all elements of the linked list into the stack |
|  | | Node node = head; |
|  | | while (node != null) { |
|  | | s.push(node.data); |
|  | | node = node.next; |
|  | | } |
|  | |  |
|  | | // traverse the linked list again |
|  | | node = head; |
|  | | while (node != null) |
|  | | { |
|  | | // pop the top element from the stack |
|  | | int top = s.pop(); |
|  | |  |
|  | | // compare the popped element with current node's data |
|  | | // return false if mismatch happens |
|  | | if (top != node.data) { |
|  | | return false; |
|  | | } |
|  | |  |
|  | | // advance to the next node |
|  | | node = node.next; |
|  | | } |
|  | |  |
|  | | // we reach here only when the linked list is palindrome |
|  | | return true; |
|  | | } |
|  | |  |
|  | | public static void main(String[] args) |
|  | | { |
|  | | Node head = new Node(1); |
|  | | head.next = new Node(2); |
|  | | head.next.next = new Node(3); |
|  | | head.next.next.next = new Node(2); |
|  | | head.next.next.next.next = new Node(1); |
|  | |  |
|  | | if (isPalindrome(head)) { |
|  | | System.out.print("Linked List is a palindrome."); |
|  | | } else { |
|  | | System.out.print("Linked List is not a palindrome."); |
|  | | } |
|  | | } |
|  | | } |