DAILY ONLINE ACTIVITIES SUMMARY

Date:	19/5/2020	**Name:**	Vleena Mascarenhas
Sem & Sec	8th & B	**USN:**	4AL16CS121

Online Test Summary

Subject	Big Data Analytics		
Max. Marks	30	**Score**	12

Certification Course Summary

Course	Introduction to Ethical Hacking		
Certificate Provider	Great learning academy	**Duration**	6hrs

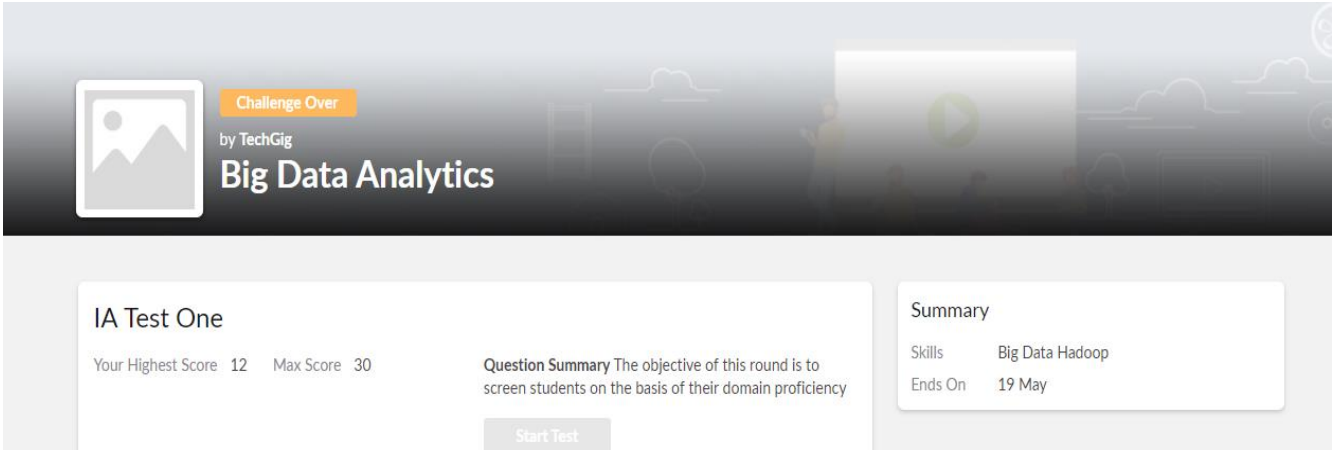
Coding Challenges

Problem Statement:
1.To find shortest palindrome string
2.write a simple code to identify given linked list is palindrome or not by using stack

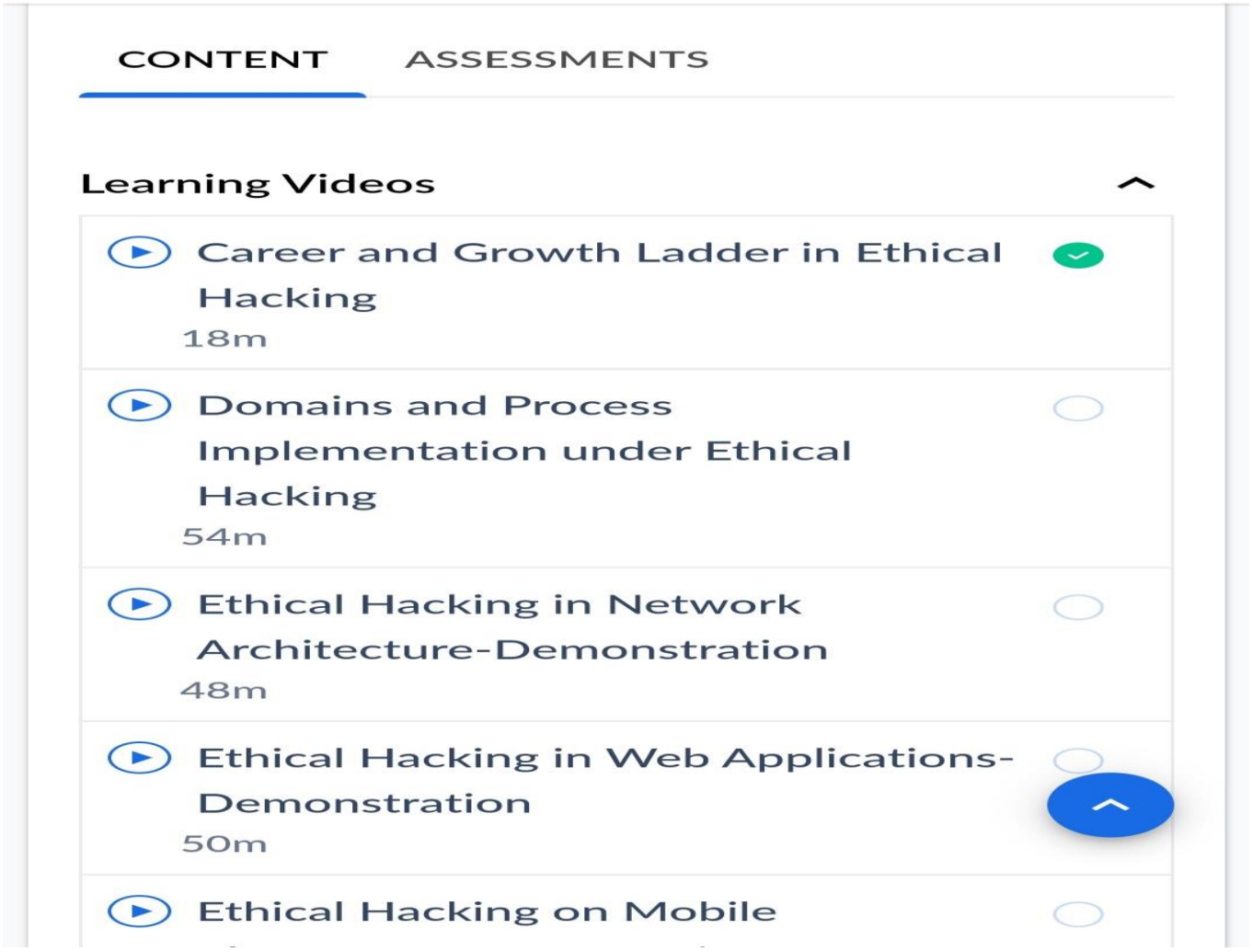
Status: Solved

Uploaded the report in Github	yes
If yes Repository name	vleena
Uploaded the report in slack	yes

Online Test Details: (Attach the snapshot and briefly write the report for the same)



Certification Course Details: (Attach the snapshot and briefly write the report for the same)



Coding Challenges Details: (Attach the snapshot and briefly write the report for the same)

```
1.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));

}
```

2.

```
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 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.");
        }
}
}
```