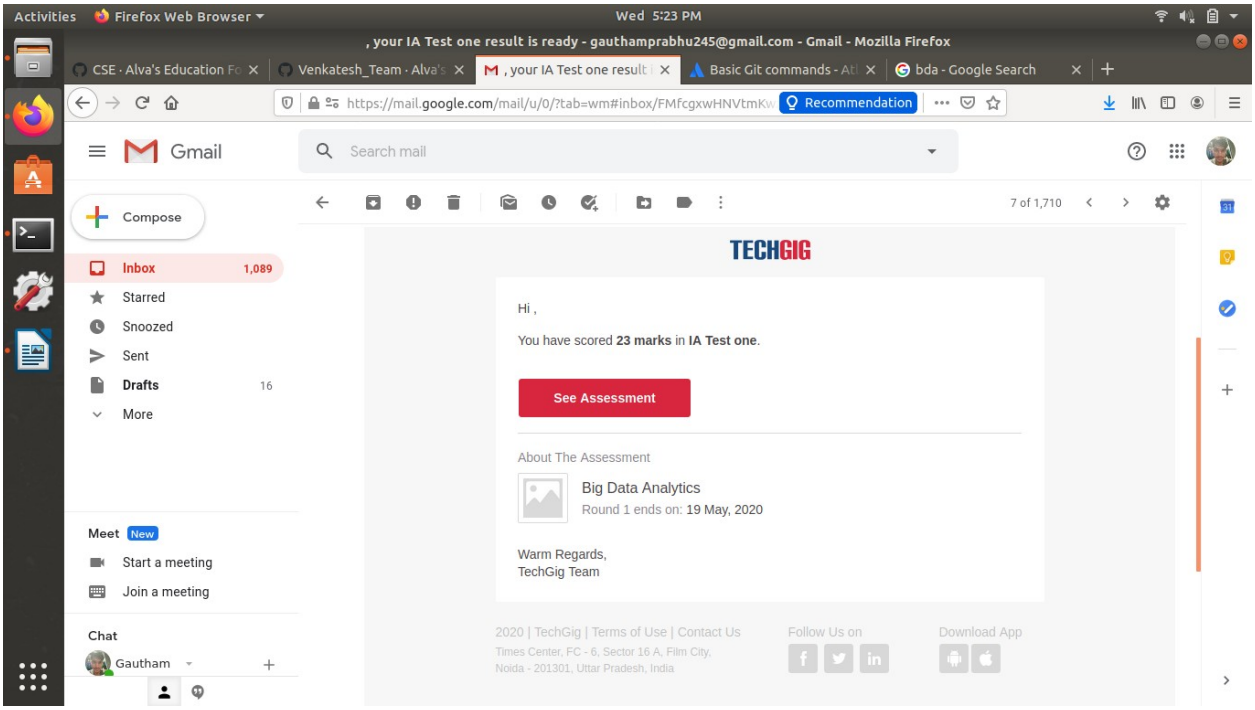
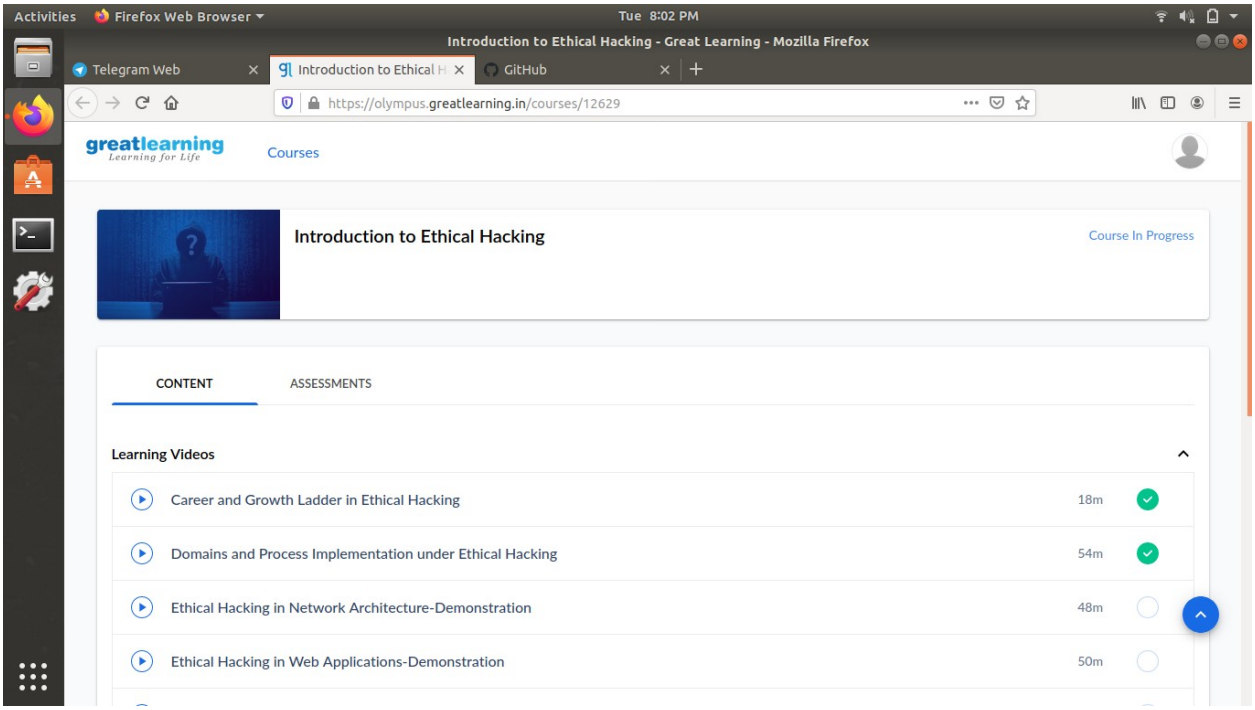
DAILY ONLINE ACTIVITIES SUMMARY

Date:	**19/5/2020**	**Name:**	**Gautham Prabhu**
Sem & Sec	**8th Sem**	**USN:**	**4AL16CS035**
Online Test Summary			
Subject	**Big Data Analytics**		
Max. Marks	**30**	**Score**	**23**
Certification Course Summary			
Course	**Introduction to Ethical Hacking**		
Certificate Provider	**greatlearning.in**	**Duration**	**6 hrs**
Coding Challenges			
Problem Statement: 1)Add letters to given letter/word and fine the shortest palindrome.2) To check if given linked list is palindrome or not			
Status: Completed			
Uploaded the report in Github	**Yes**		
If yes Repository name	**Daily_report**		
Uploaded the report in slack	**yes**		

Online Test Details:



Certification Course Details:



Coding Challenges Details:

Program 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));

}
```

Program 2

```
import java.util.Stack;

class Node {
int data;
Node next;

Node(int i)
{
	this.data = i;
	this.next = null;
}
};

class Main
{
public static boolean isPalindrome(Node head)
{
// construct an empty stack
Stack s = new Stack<>();
Node node = head;
while (node != null) {
s.push(node.data);
node = node.next;
}

	node = head;
	while (node != null)
	{
		int top = s.pop();
		if (top != node.data) {
			return false;
		}
		node = node.next;
	}

	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.");
        }
}
}
```

Program 3

```
#include<stdio.h>

#include<math.h>


int max(int num1, int num2)

{

   return (num1 > num2 ) ? num1 : num2;

}


int countWays(int arr[], int n)

   {

      int max_val = 0;

      for (int i = 0; i < n; i++)

         max_val = max(max_val, arr[i]);


      int freq[20]={0};

      for (int i = 0; i < n; i++)

         freq[arr[i]]++;


      int ans = 0;
```

```
    ans += freq[0] * (freq[0] - 1) * (freq[0] - 2) / 6;

    for (int i = 1; i <= max_val; i++)
        ans += freq[0] * freq[i] * (freq[i] - 1) / 2;

    for (int i = 1; 2 * i <= max_val; i++)
        ans += freq[i] * (freq[i] - 1) / 2 * freq[2 * i];

    for (int i = 1; i <= max_val; i++) {
        for (int j = i + 1; i + j <= max_val; j++)
            ans += freq[i] * freq[j] * freq[i + j];
    }

    return ans;
}

int main()
{
    int arr[]={1, 5, 3, 2 };
    int n = sizeof(arr)/sizeof(int);
    printf("%d",countWays(arr, n));
    return 0;
}
```