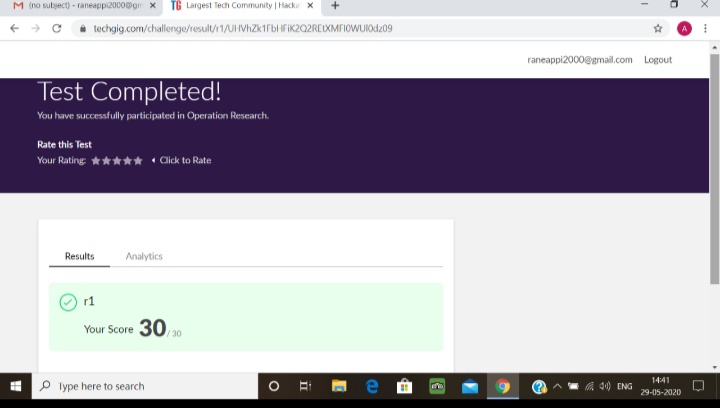
**DAILY ONLINE ACTIVITIES SUMMARY**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date:** | **29-05-2020** | | | | | **Name:** | **Apeksha Rane** | |
| **Sem & Sec** | **A** | | | | | **USN:** | **4AL17CS010** | |
| **Online Test Summary** | | | | | | | | |
| **Subject** | | **Operating Research.** | | | | | | |
| **Max. Marks** | | **30** | | **Score** | | | **30** | |
| **Certification Course Summary** | | | | | | | | |
| **Course** | **INTRODUCTION TO CLOUD** | | | | | | | |
| **Certificate Provider** | | | COGNITIVE  CLASS IBM. | | **Duration** | | | 4hours |
| **Coding Challenges** | | | | | | | | |
| **Problem Statement:**1. Python program to calculate the number of lowercase and uppercase letters in a string  2. We are given 3 strings: str1, str2, and str3. Str3 is said to be a shuffle of str1 and str2 if it can be formed by interleaving the characters of str1 and str2 in a way that maintains the left to right ordering of the characters from each string.  3. c program to solve a system of linear congruences by applying the Chinese Remainder Theorem.  4. Write a Java Program to check whether the given number is Armstrong number or not  5. Write a Java Program to find longest substring without repeating characters in a string. | | | | | | | | |
| **Status: Done** | | | | | | | | |
| **Uploaded the report in Github** | | | | | **YES** | | | |
| **If yes Repository name** | | | | | <https://github.com/Apeksha12appu/19-5-2020-online-coding-activity> | | | |
| **Uploaded the report in slack** | | | | | **YES** | | | |

Online Test Details:

Subject:-Operating Research.

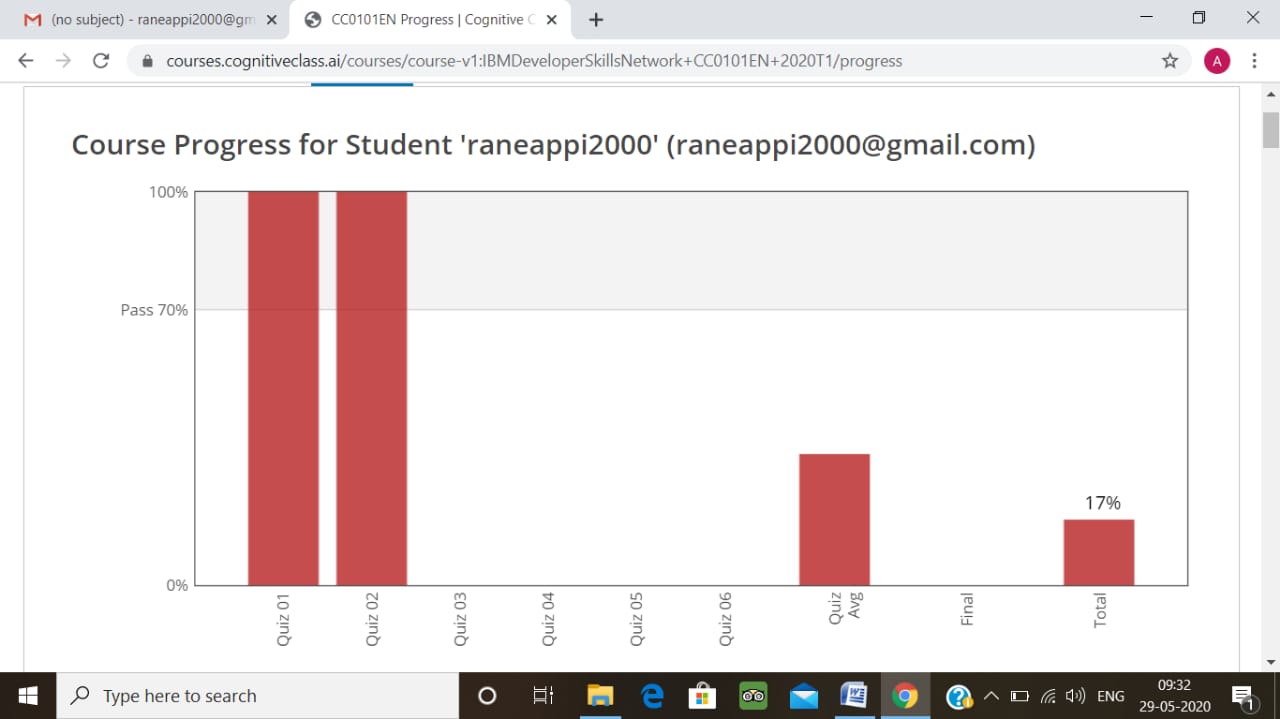


Certification Course Details:

**Introduction to Cloud:**

Today I have studied Cloud **Adoption and Emerging Technologies** **:**

* Introduction and Objectives.
* Cloud Adoption-Some Case Studies
* Internet of Things in the Cloud
* Artificial intelligence on the cloud
* Block chain and Analytics on the Cloud



Coding Challenges Details:

1. Python program to calculate the number of lowercase and uppercase letters in a string

Description:  
Take a string as input and find the number of uppercase and lower case letters in the string and print the count.  
Note: any spaces has to be ignored  
Eg: string is : 'This is Python'  
Upper case characters : 2  
Lower case characters : 10

def upperlower(string):

upper = 0

lower = 0

for i in range(len(string)):

if (ord(string[i]) >= 97 and

ord(string[i]) <= 122):

lower += 1

elif (ord(string[i]) >= 65 and

ord(string[i]) <= 90):

upper += 1

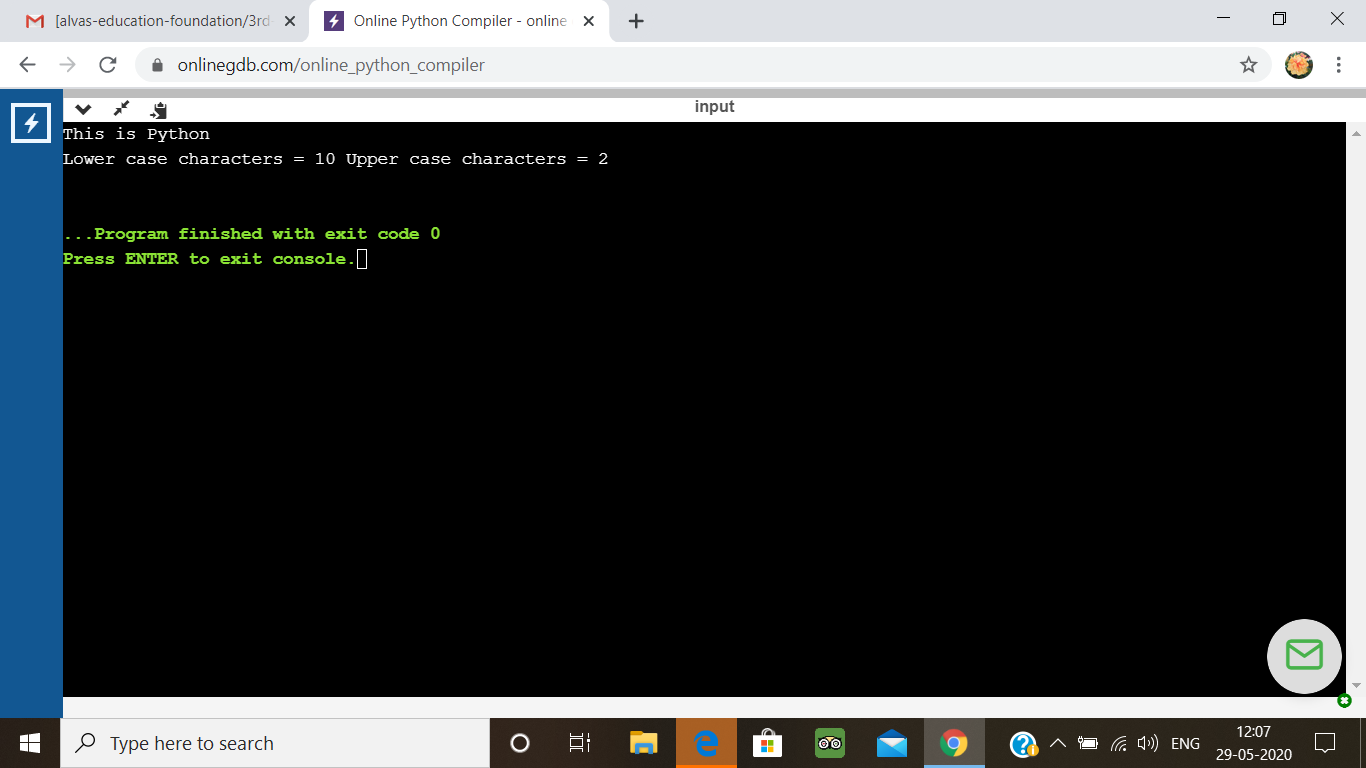
print('Lower case characters = %s' %lower,

'Upper case characters = %s' %upper)

string = input()

upperlower(string)

**output:**



2. We are given 3 strings: str1, str2, and str3. Str3 is said to be a shuffle of str1 and str2 if it can be formed by interleaving the characters of str1 and str2 in a way that maintains the left to right ordering of the characters from each string. For example, given str1="abc" and str2="def", str3="dabecf" is a valid shuffle since it preserves the character ordering of the two strings. So, given these 3 strings write a function that detects whether str3 is a valid shuffle of str1 and str2.

public class Main{

static boolean isInterleaved (String A, String B, String C)

{

int i = 0, j = 0, k = 0;

while (k != C.length())

{

if (i<A.length()&&A.charAt(i) == C.charAt(k))

i++;

else if (j<B.length()&&B.charAt(j) == C.charAt(k))

j++;

else

return false;

k++;

}

if (i < A.length() || j < B.length())

return false;

return true;

}

public static void main(String []args){

String A = "abc";

String B = "def";

String C = "dabecf";

if (isInterleaved(A, B, C) == true)

System.out.printf("%s is interleaved of %s and %s", C, A, B);

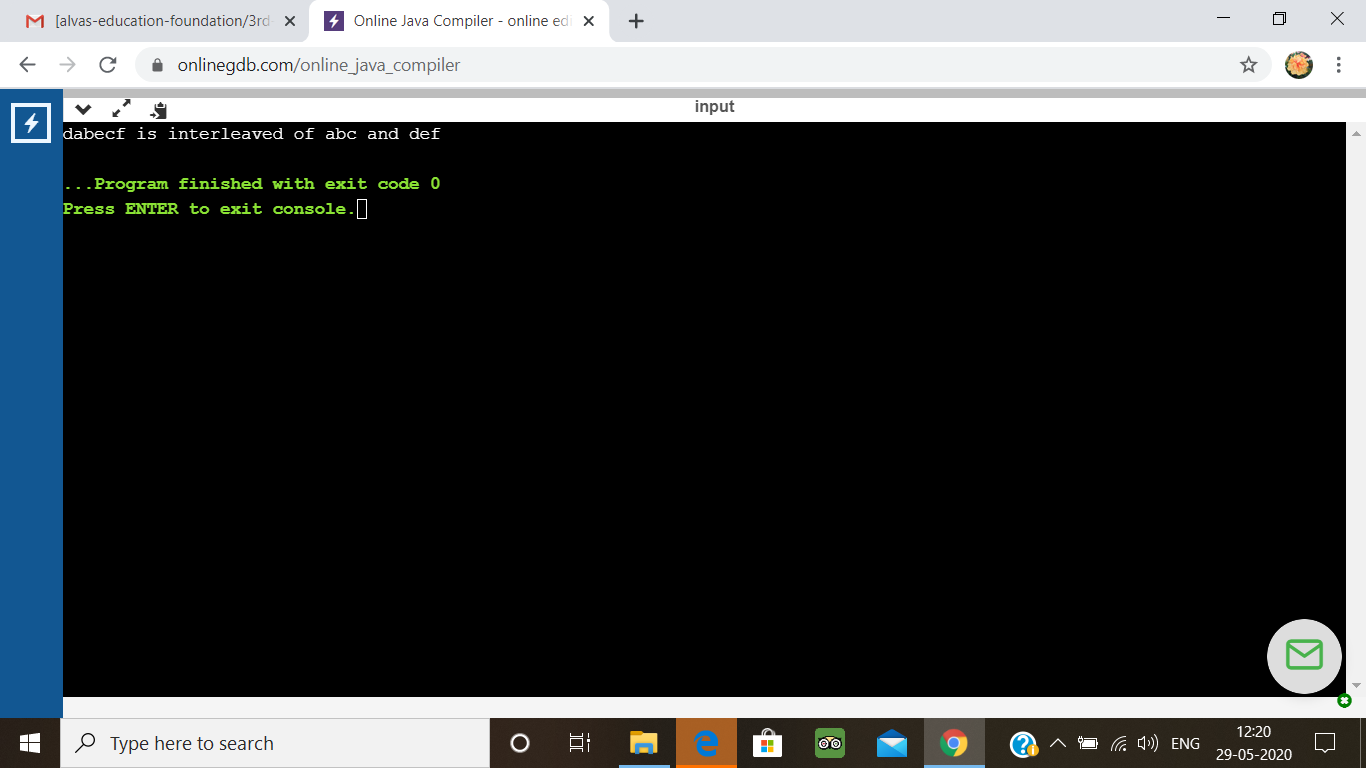
else

System.out.printf("%s is not interleaved of %s and %s", C, A, B);

}

}

**Output:**



3.write a c program to solve a system of linear congruences by applying the Chinese Remainder Theorem.

#include <stdio.h>

int mul\_inv(int a, int b)

{

int b0 = b, t, q;

int x0 = 0, x1 = 1;

if (b == 1) return 1;

while (a > 1) {

q = a / b;

t = b, b = a % b, a = t;

t = x0, x0 = x1 - q \* x0, x1 = t;

}

if (x1 < 0) x1 += b0;

return x1;

}

int chinese\_remainder(int \*n, int \*a, int len)

{

int p, i, prod = 1, sum = 0;

for (i = 0; i < len; i++) prod \*= n[i];

for (i = 0; i < len; i++) {

p = prod / n[i];

sum += a[i] \* mul\_inv(p, n[i]) \* p;

}

return sum % prod;

}

int main(void)

{

int n[] = { 3, 5, 7 };

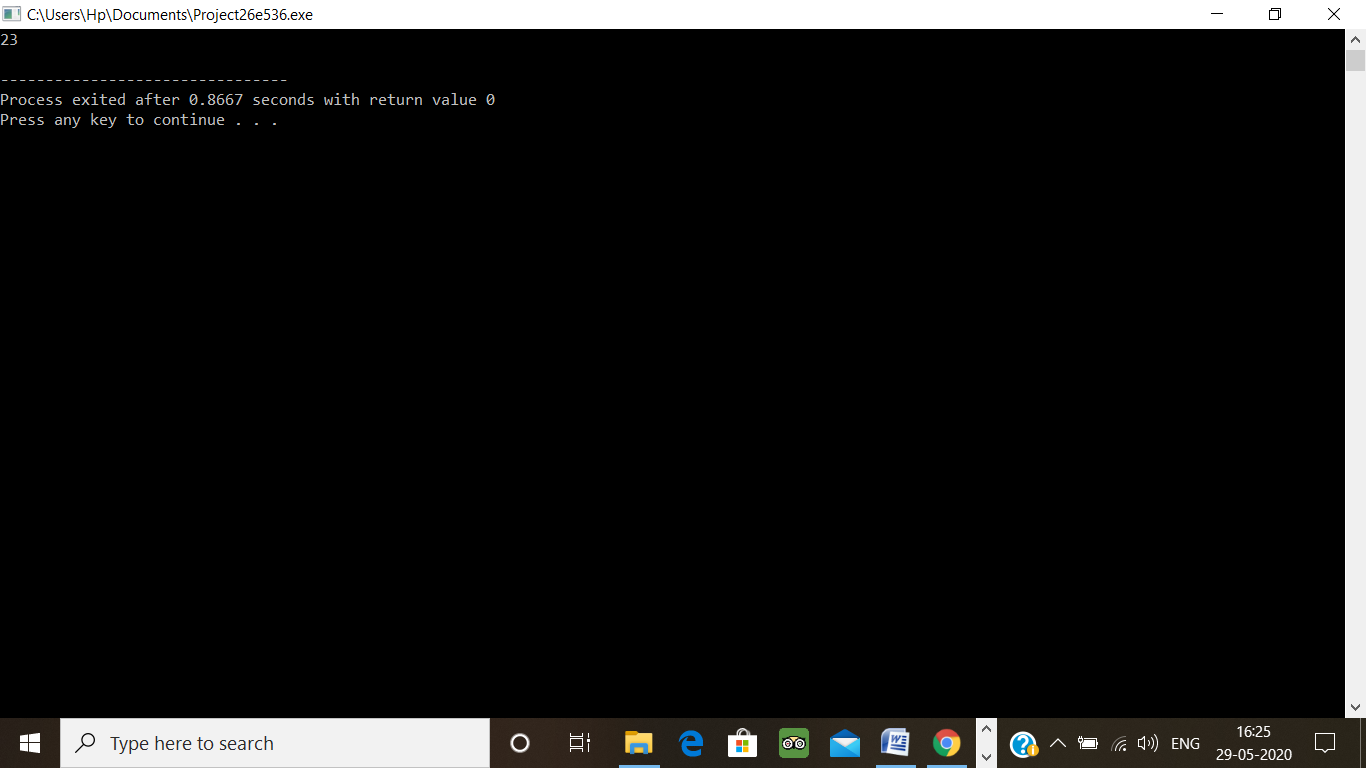
int a[] = { 2, 3, 2 };

printf("%d\n", chinese\_remainder(n, a, sizeof(n)/sizeof(n[0])));

return 0;

}

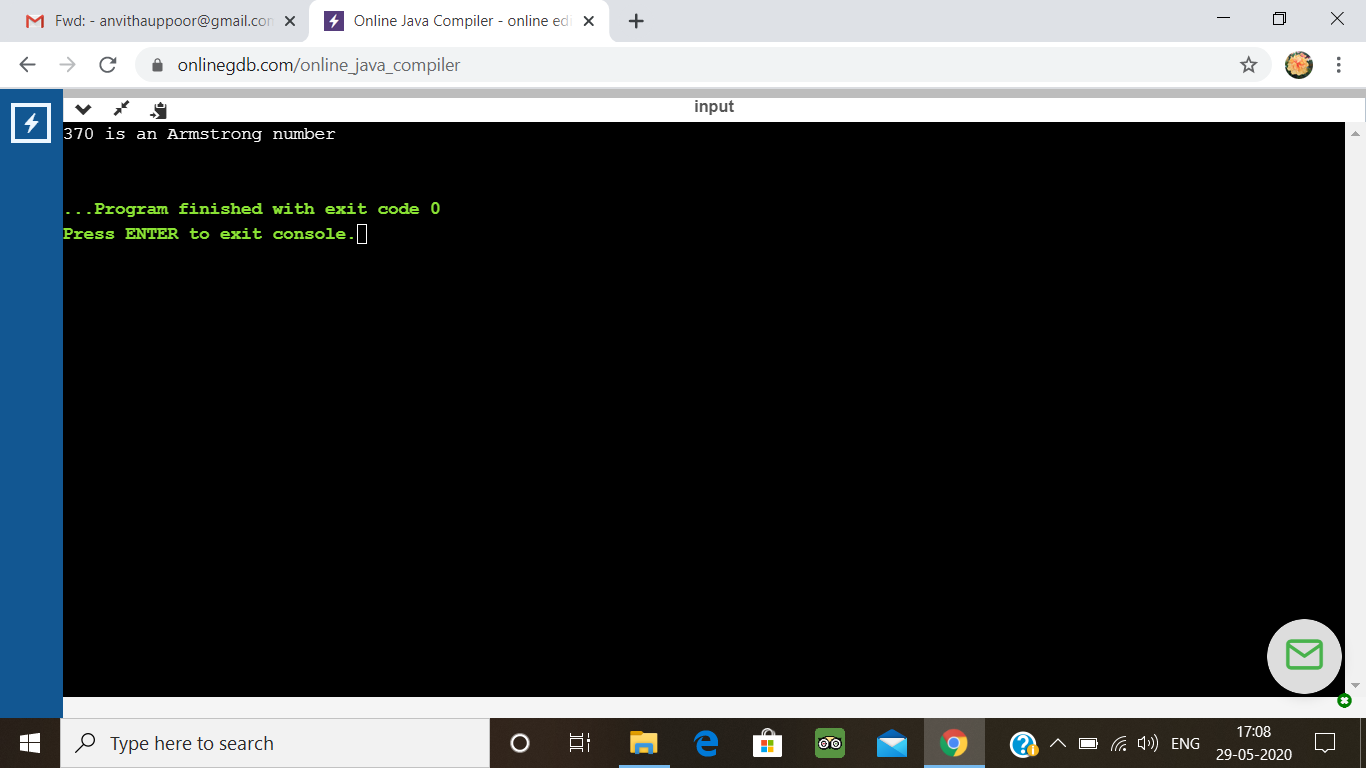
**Output:**



4. Write a Java Program to check whether the given number is Armstrong number or not

public class Main {  
  
    public static void main(String[] args) {  
  
        int num = 370, number, temp, total = 0;  
  
        number = num;  
        while (number != 0)  
        {  
            temp = number % 10;  
            total = total + temp\*temp\*temp;  
            number /= 10;  
        }  
        if(total == num)  
            System.out.println(num + " is an Armstrong number");  
        else  
            System.out.println(num + " is not an Armstrong number");  
    }  
}

**output:**



5. Write a Java Program to find longest substring without repeating characters in a string.

import java.util.LinkedHashMap;

public class Main

{

static void longestSubstring(String inputString)

{

char[] charArray = inputString.toCharArray();

String longestSubstring = null;

int longestSubstringLength = 0;

LinkedHashMap<Character, Integer> charPosMap = new LinkedHashMap<Character, Integer>();

for (int i = 0; i < charArray.length; i++)

{

char ch = charArray[i];

if(!charPosMap.containsKey(ch))

{

charPosMap.put(ch, i);

}

else

{

i = charPosMap.get(ch);

charPosMap.clear();

}

if(charPosMap.size() > longestSubstringLength)

{

longestSubstringLength = charPosMap.size();

longestSubstring = charPosMap.keySet().toString();

}

}

System.out.println("Input String : "+inputString);

System.out.println("The longest substring : "+longestSubstring);

System.out.println("The longest Substring Length : "+longestSubstringLength);

}

public static void main(String[] args)

{

longestSubstring("javaconceptoftheday");

}

}

**Output:**

