# **Format for uploading details in GitHub and Slack in word file format**

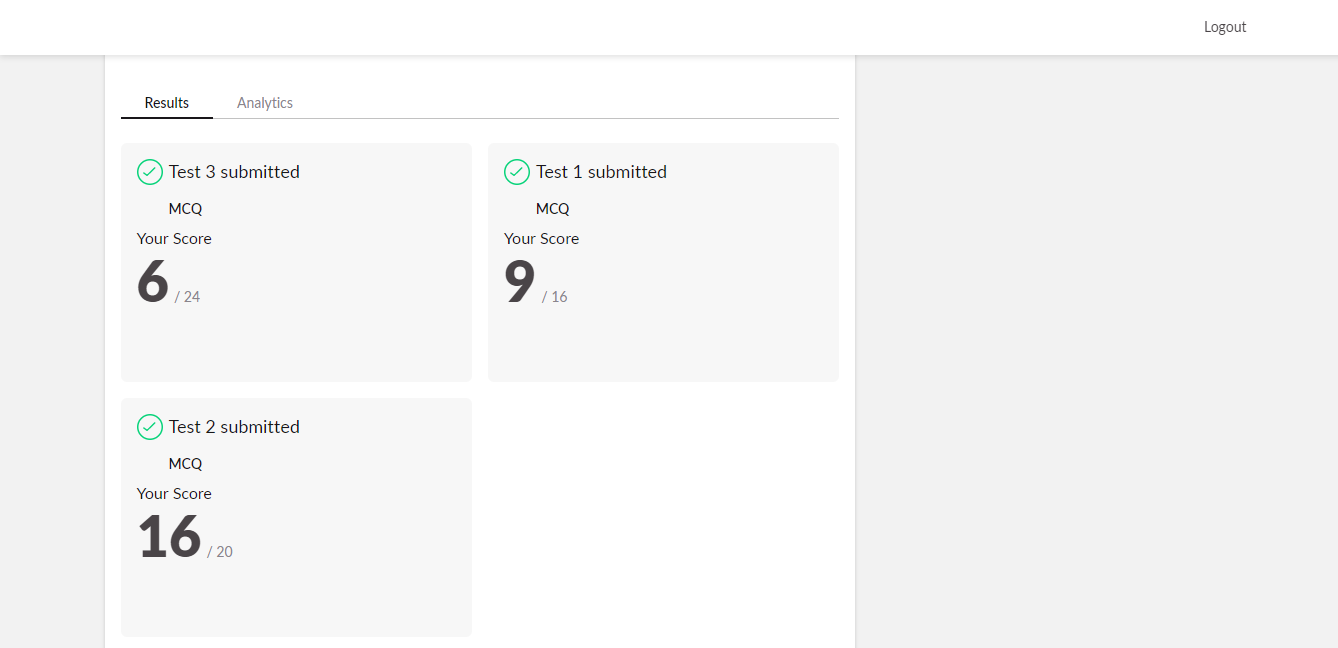
**Student Name: Shrinivasa**

**Class and Sec: VI B**

**USN: 4AL17CS092**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Online Test Details** | | | | |
| **Subject** | **Cryptography Network Security and Cyber Laws** | | | |
| **Semester** | **VI - B** | | **Duration** | **30 Minutes** |
| **51%** | | **31/60** | | |

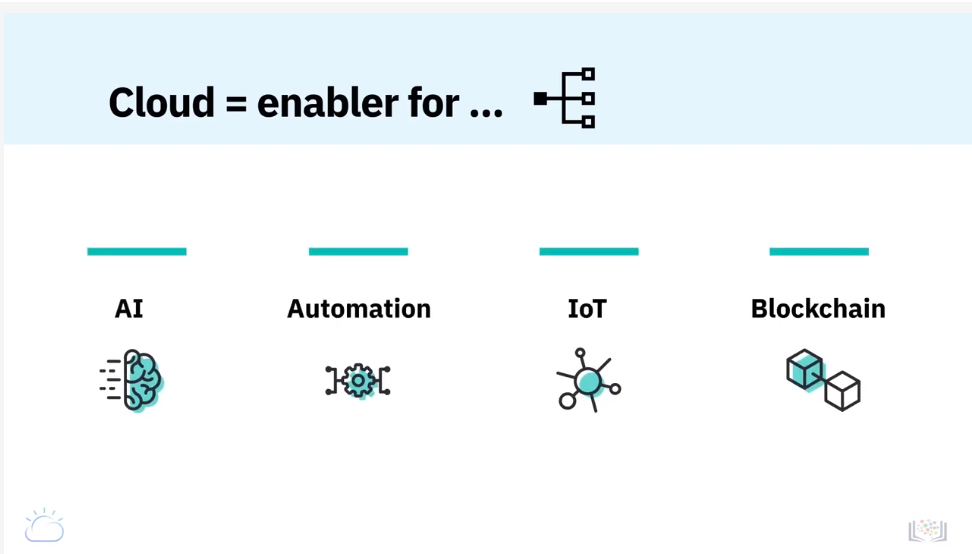
**Encl: snapshot of the test result**

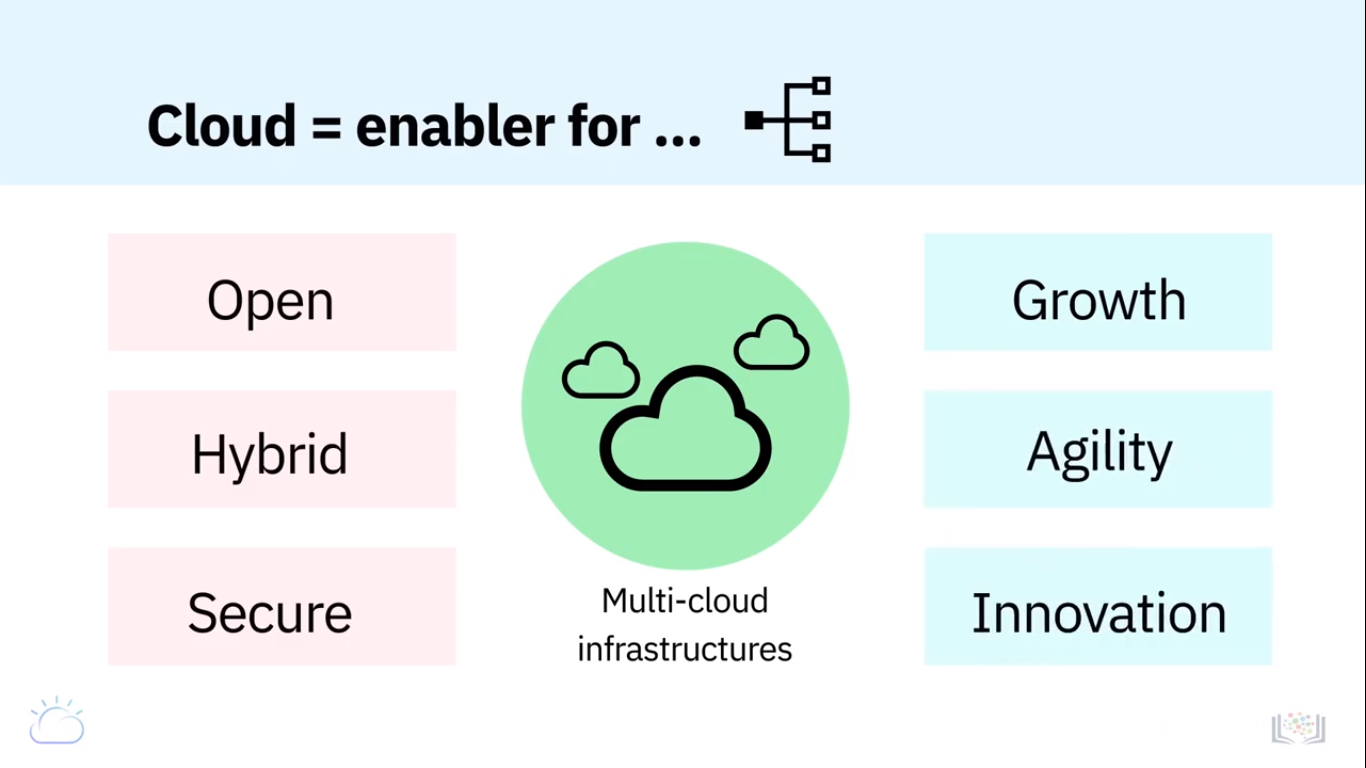


|  |  |  |  |
| --- | --- | --- | --- |
| **Certification Course Details** | | | |
| **Course** | **Introduction to Cloud** | | |
| **Certificate Provider** | **Cognitiveclass.ai** | **Duration** | **6 hours** |

**Encl: snapshots of the daily class activities (at least two snap shots)**

**Progress on 08-06-2020**







|  |  |
| --- | --- |
| **Coding Challenges** | |
| **Problem Statement: Pro1(c++), Pro2(c), Pro3(java), Pro4(python).** | |
| **Status: Completed** | |
| **Uploaded the report both in GitHub & Slack** | **Yes** |

**Encl: snapshots of your response to challenge.**

[**https://github.com/Shrinivasakunder/certification-and-online-coding/tree/master/Online%20coding**](https://github.com/Shrinivasakunder/certification-and-online-coding/tree/master/Online%20coding)

**1.Write C++ program to Check whether a number can be represented as difference of two squares.**

#include <bits/stdc++.h>

using namespace std;

bool difSquare(int n)

{

if (n % 4 != 2) {

return true;

}

return false;

}

int main()

{

int n;

std::cout<<"enter the number: ";

std::cin>>n;

if (difSquare(n)) {

cout << "Yes\n";

}

else {

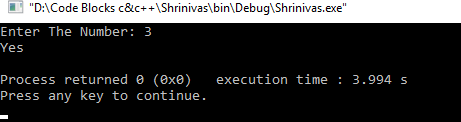
cout << "No\n";

}

return 0;

}

**Output:**



**2.C Program to Generate All the Set Partitions of n Numbers Beginning from 1 and so on.**

#include <stdio.h>

#include <stdlib.h>

typedef struct {

int first;

int n;

int level;

} Call;

void print(int n, int \* a) {

int i ;

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

printf("%d", a[i]);

}

printf("\n");

}

void integerPartition(int n, int \* a){

int first;

int i;

int top = 0;

int level = 0;

Call \* stack = (Call \* ) malloc (sizeof(Call) \* 1000);

stack[0].first = -1;

stack[0].n = n;

stack[0].level = level;

while (top >= 0){

first = stack[top].first;

n = stack[top].n;

level = stack[top].level;

if (n >= 1) {

if (first == - 1) {

a[level] = n;

print(level, a);

first = (level == 0) ? 1 : a[level-1];

i = first;

} else {

i = first;

i++;

}

if (i <= n / 2) {

a[level] = i;

stack[top].first = i;

top++;

stack[top].first = -1;

stack[top].n = n - i;

stack[top].level = level + 1;

} else {

top--;

}

} else {

top --;

}

}

}

int main(){

int N = 1;

int \* a = (int \* ) malloc(sizeof(int) \* N);

int i;

printf("\nEnter a number N to generate all set partition from 1 to N: ");

scanf("%d", &N);

for ( i = 1; i <= N; i++)

{

printf("\nInteger partition for %d is: \n", i);

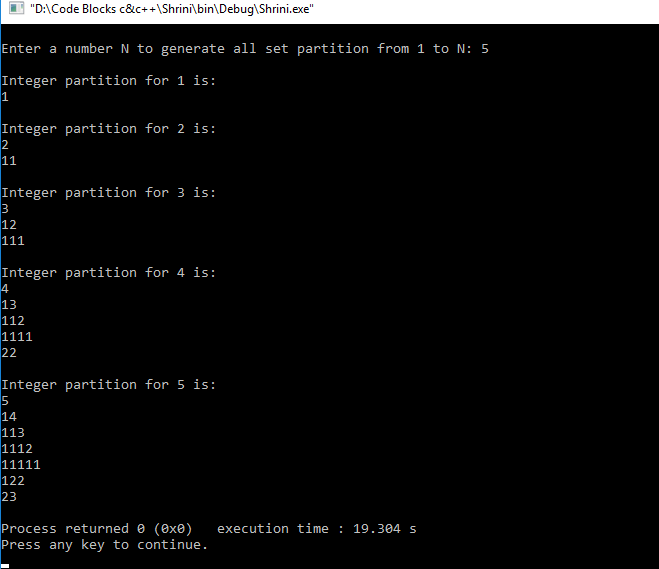
integerPartition (i, a);

}

return(0);

}

**Output:**



**3.Java program to delete a node from the middle of the singly linked list .**

**package** pblm;

**public** **class** pro2{

**class** Node{

**int** data;

Node next;

**public** Node(**int** data)

{

**this**.data = data;

**this**.next = **null**;

}

}

**public** Node head = **null**;

**public** Node tail = **null**;

**public** **int** size;

**public** **void** addNode(**int** data) {

Node newNode = **new** Node(data);

**if**(head == **null**) {

head = newNode;

tail = newNode;

}

**else** {

tail.next = newNode;

tail = newNode;

}

size++;

}

**void** deleteFromMid() {

Node temp, current;

**if**(head == **null**) {

System.***out***.println("List is empty");

**return**;

}

**else** {

**int** count = (size % 2 == 0) ? (size/2) : ((size+1)/2);

**if**( head != tail ) {

temp = head;

current = **null**;

**for**(**int** i = 0; i < count-1; i++){

current = temp;

temp = temp.next;

}

**if**(current != **null**) {

current.next = temp.next;

temp = **null**;

}

**else** {

head = tail = temp.next;

temp = **null**;

}

}

**else** {

head = tail = **null**;

}

}

size--;

}

**public** **void** display() {

Node current = head;

**if**(head == **null**) {

System.***out***.println("List is empty");

**return**;

}

**while**(current != **null**) {

System.***out***.print(current.data + " ");

current = current.next;

}

System.***out***.println();

}

**public** **static** **void** main(String[] args) {

pro2 sList = **new** pro2();

sList.addNode(1);

sList.addNode(2);

sList.addNode(3);

sList.addNode(4);

System.***out***.println("Original List: ");

sList.display();

**while**(sList.head != **null**) {

sList.deleteFromMid();

System.***out***.println("Updated List: ");

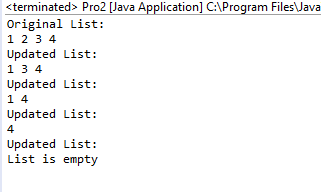
sList.display();

}

}

}

**Output:**



**4.Write a python function that will take a string and checks whether it is a palindrome or not. Return If it a palindrome, print true else print false.**

def isPal(s):

return s == s[::-1]

s = input("Enter The String: ").lower()

t = isPal(s)

if t:

print("True")

else:

print("False")

**Output:**

