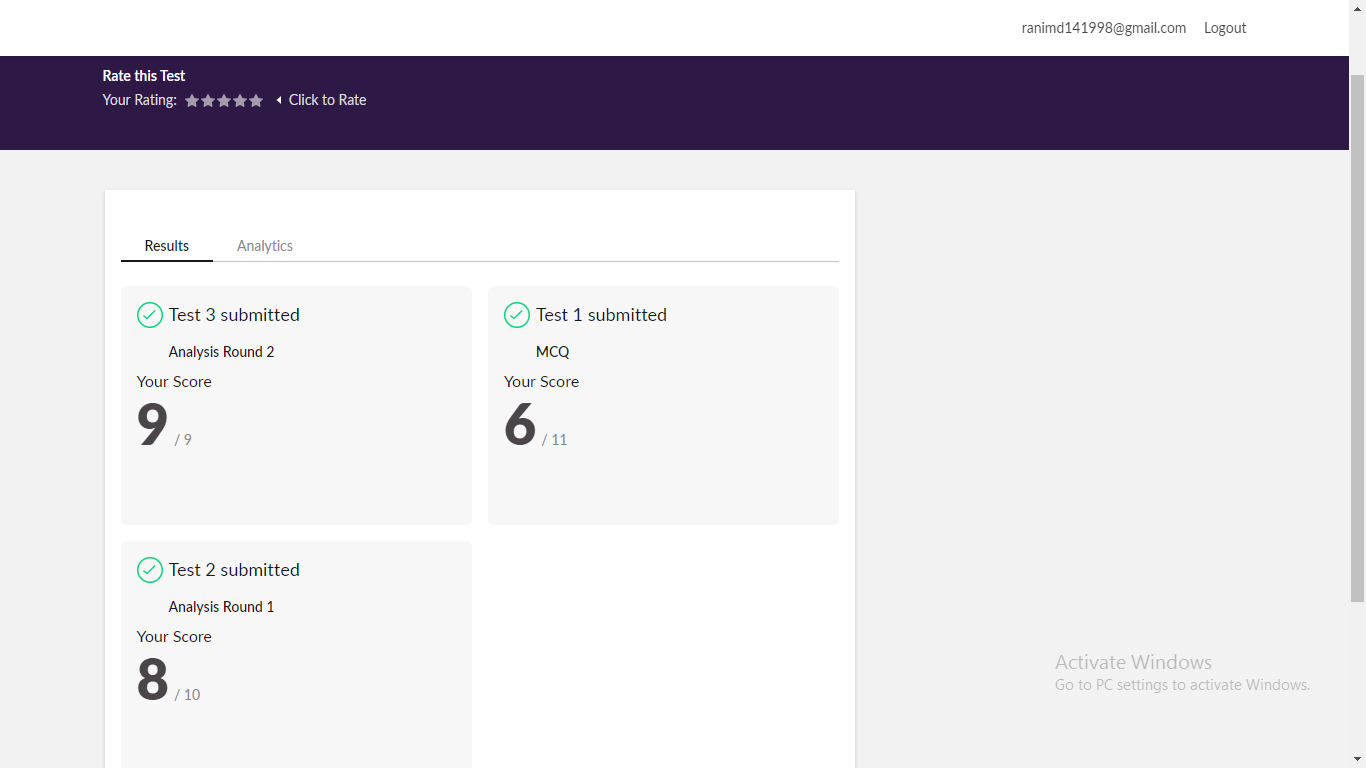
**DAILY ONLINE ACTIVITIES SUMMARY**

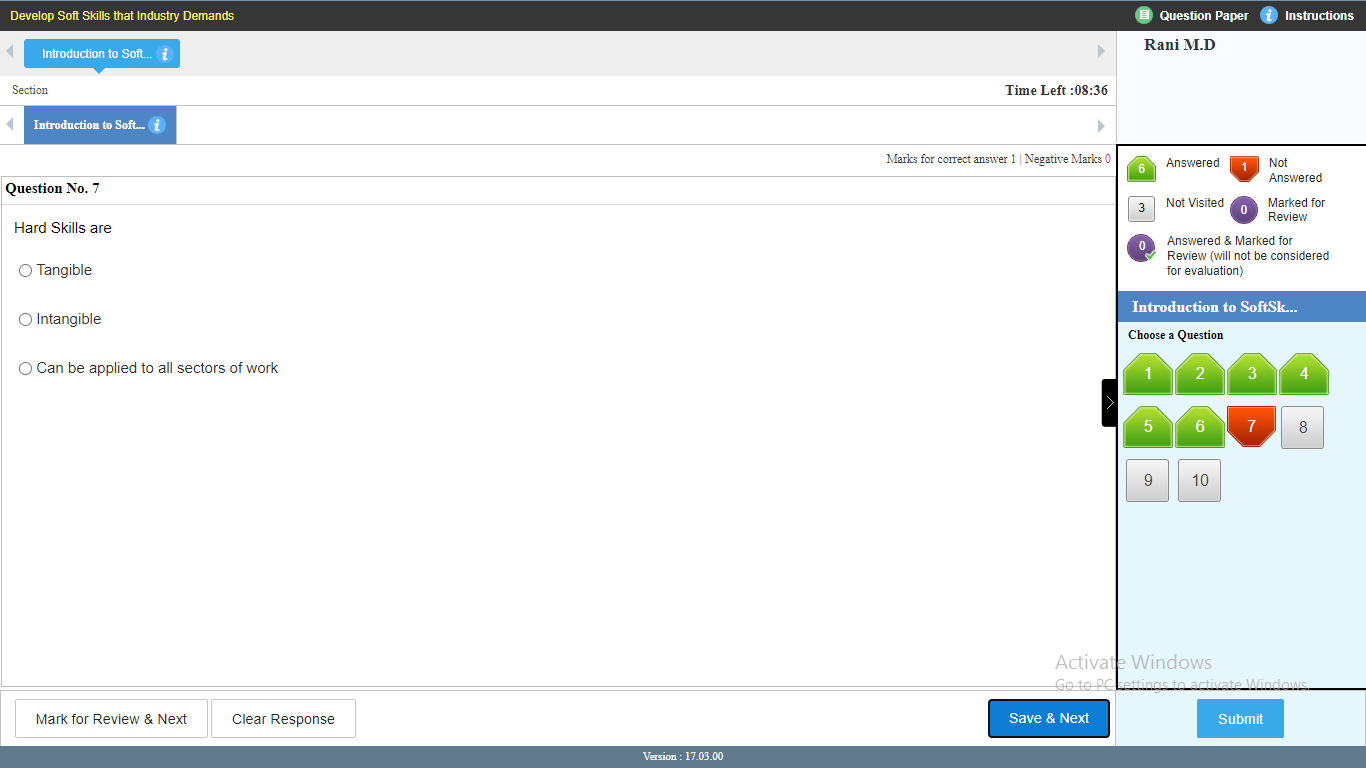
|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date:** | **08-06-2020** | | | | | **Name:** | **Rani M.D** | |
| **Sem & Sec** | **B** | | | | | **USN:** | **4AL17CS075** | |
| **Online Test Summary** | | | | | | | | |
| **Subject** | | **CNSC** | | | | | | |
| **Max. Marks** | | **60** | | **Score** | | | **20** | |
| **Certification Course Summary** | | | | | | | | |
| **Course** | **TCS edge carrier** | | | | | | | |
| **Certificate Provider** | | | TCS | | **Duration** | | | 15 days |
| **Coding Challenges** | | | | | | | | |
| **Status: Done** | | | | | | | | |
| **Uploaded the report in Github** | | | | | **YES** | | | |
| **If yes Repository name** | | | | | Daily Status | | | |
| **Uploaded the report in slack** | | | | | **YES** | | | |

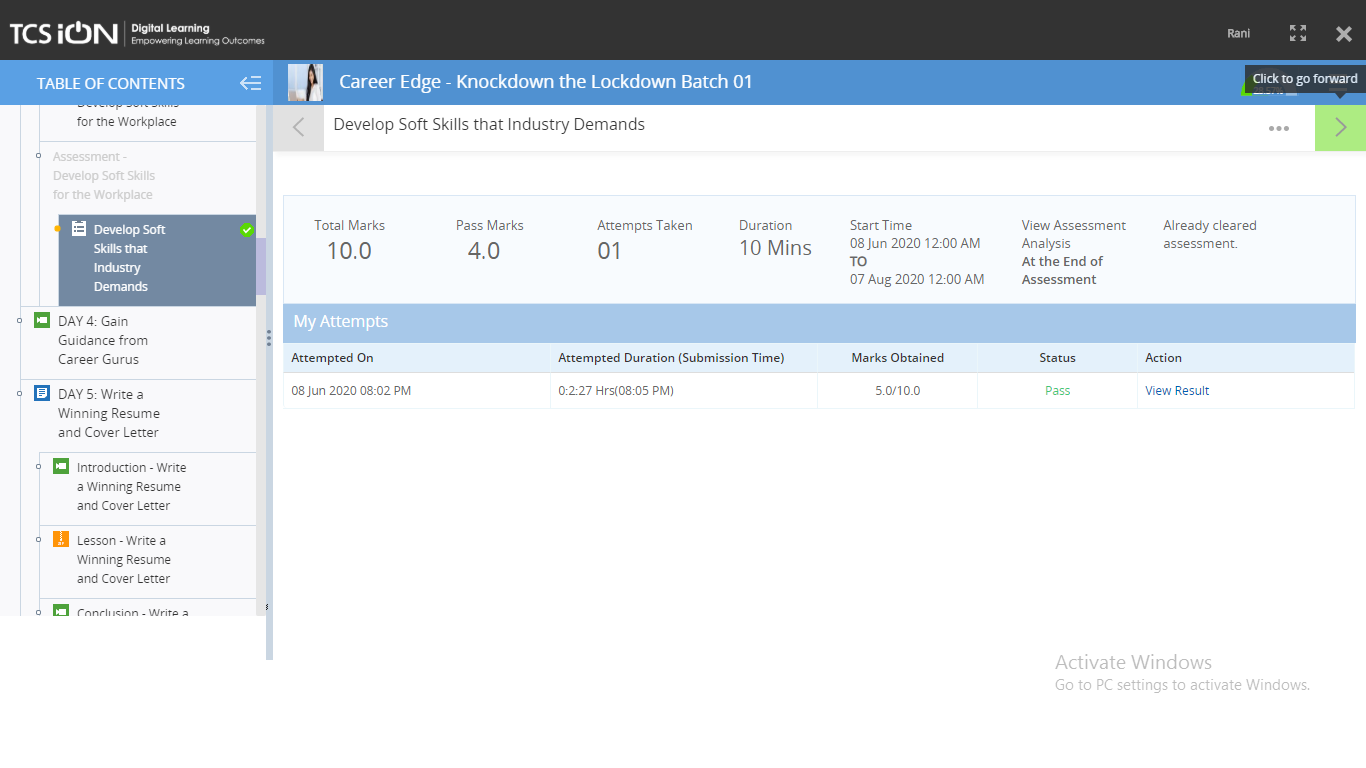
Online Test Details:

Subject:-CNSC.

CERTIFICATION COURSE:







**Coding Challenges Details:**

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

Given a number N, the task is to check if this number can be represented as the difference of two perfect squares or not.

input: N = 3  
Output: Yes  
Explanation:  
2^2 – 1^1 = 3

#include <bits/stdc++.h>

using namespace std;

bool difSquare(int n)

{

if (n % 4 != 2) {

return true;

}

return false;

}

int main()

{

int n = 3;

if (difSquare(n)) {

cout << "Yes\n";

}

else {

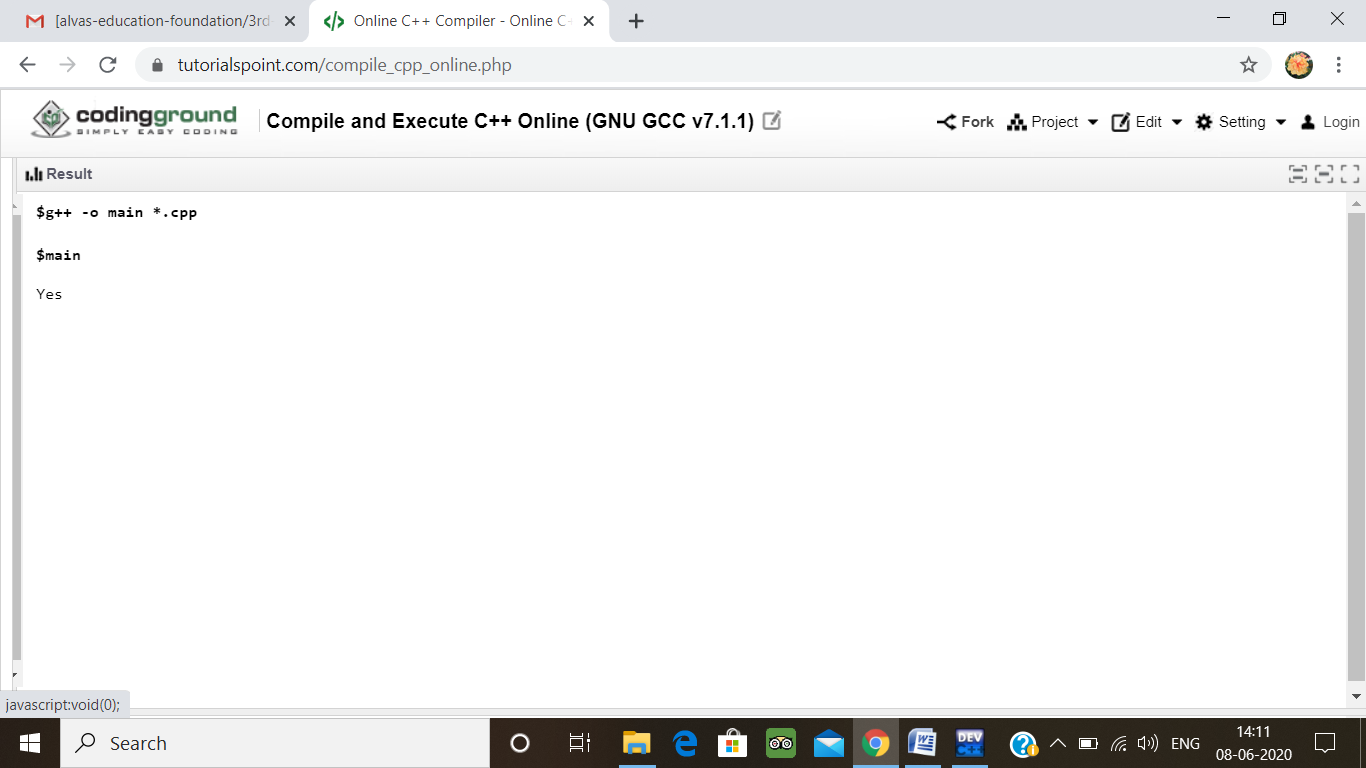
cout << "No\n";

}

return 0;

}

**Output:**



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

This algorithm partitions an integer into numbers which sum up to form the original number. It generates partitions of a set of numbers for a given range.

**Sample Input**

Enter a number N to generate all set partition from 1 to N: 5  
Integer partition for 1 is:  
1

Integer partition for 2 is:  
2  
11

Integer partition for 3 is:  
3  
12  
111

Integer partition for 4 is:  
4  
13  
112  
1111  
22

Integer partition for 5 is:  
5  
14  
113  
1112  
11111  
122  
23

#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);

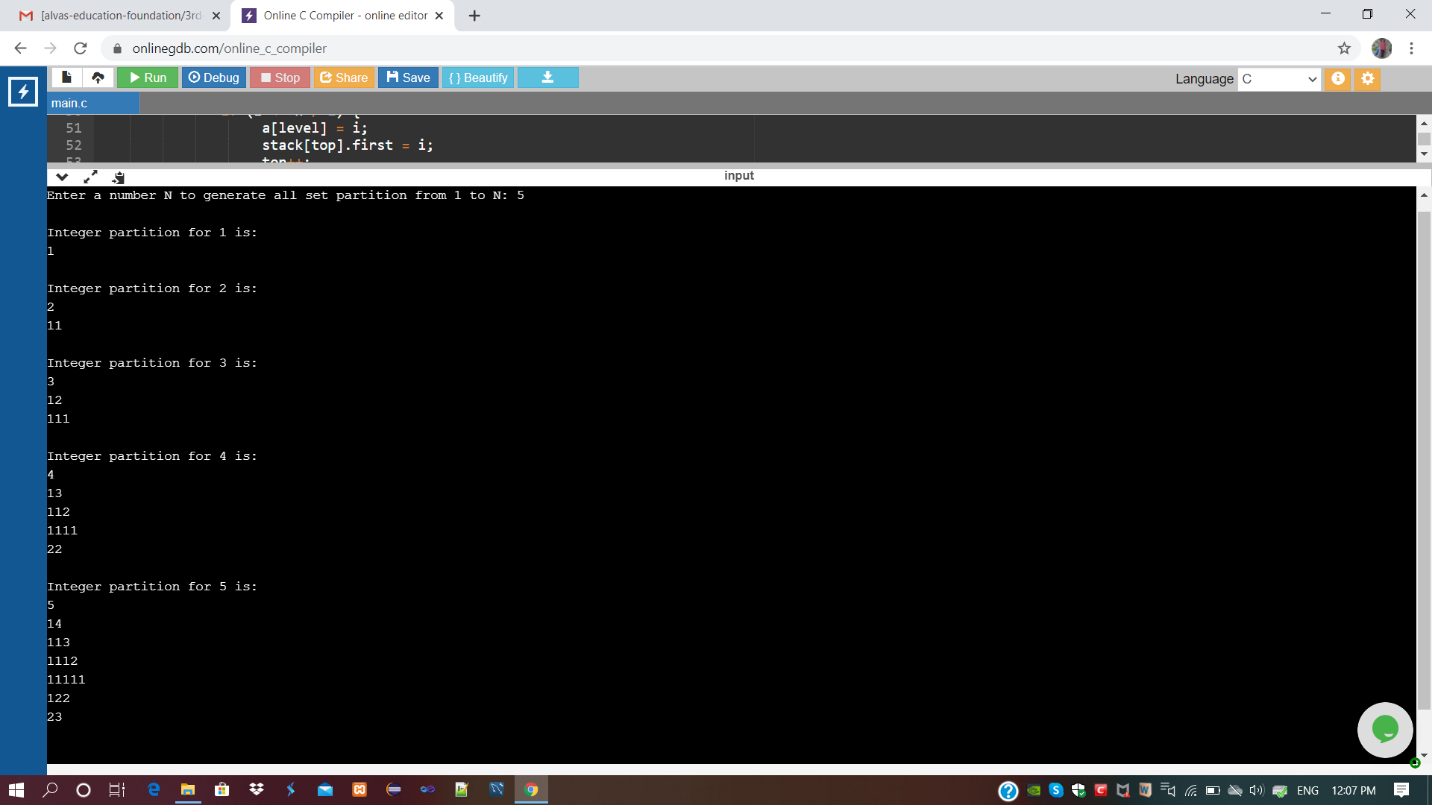
integerPartition (i, a);

}

return(0);

}

Output:



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

In this program, we will create a singly linked list and delete a node from the middle of the list. To accomplish this task, we will calculate the size of the list and then divide it by 2 to get the mid-point of the list. Node temp will point to head node. We will iterate through the list till midpoint is reached. Now, the temp will point to middle node and node current will point to node previous to temp. We delete the middle node such that current's next node will point to temp's next node.

public class deleteMid{

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) {

deleteMid sList = new deleteMid();

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: ");

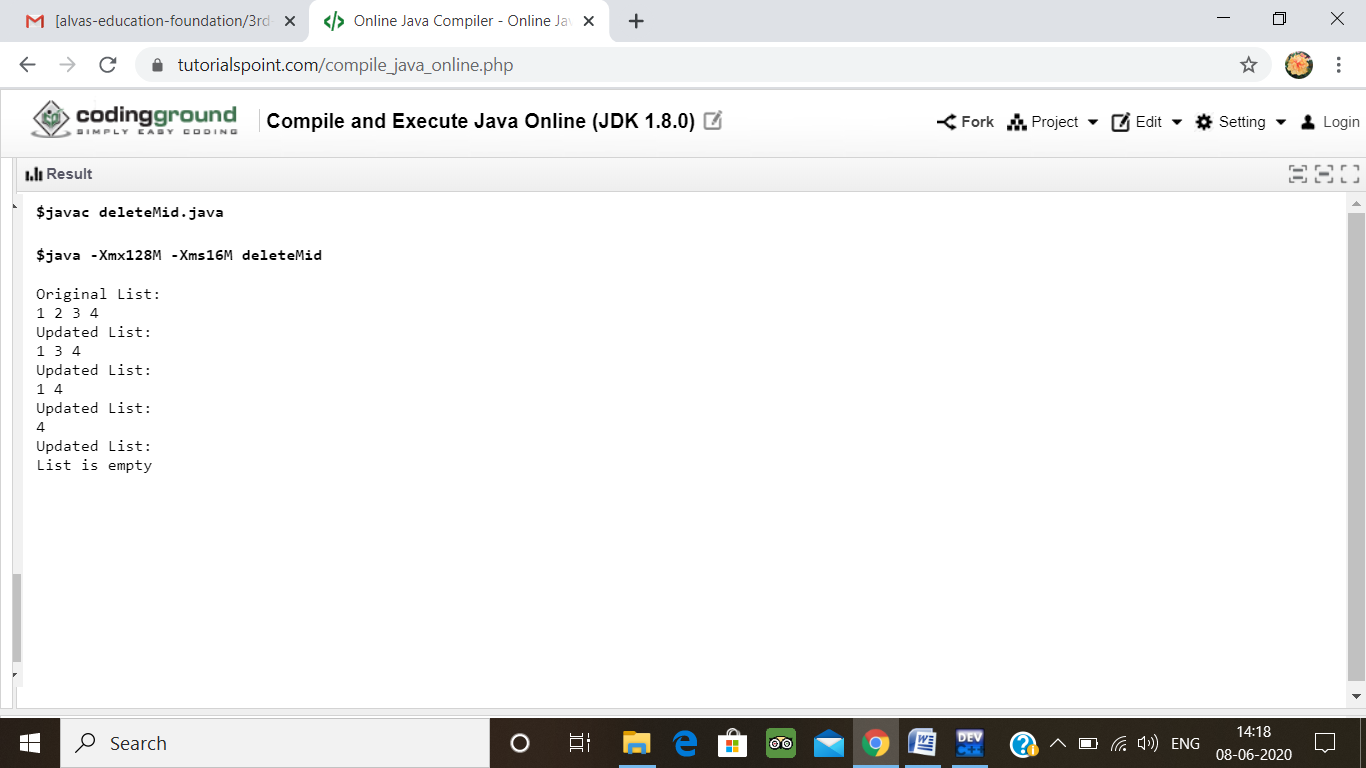
sList.display();

}

}

}

**Output:**



4.  Program program to find whether a string is a palindrome or not

Description:  
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

Eg: String is : 'aba'  
Output: True

def isPalindrome(s):

return s == s[::-1]

s = input()

ans = isPalindrome(s)

if ans:

print("Yes")

else:

print("No")

**output:**

