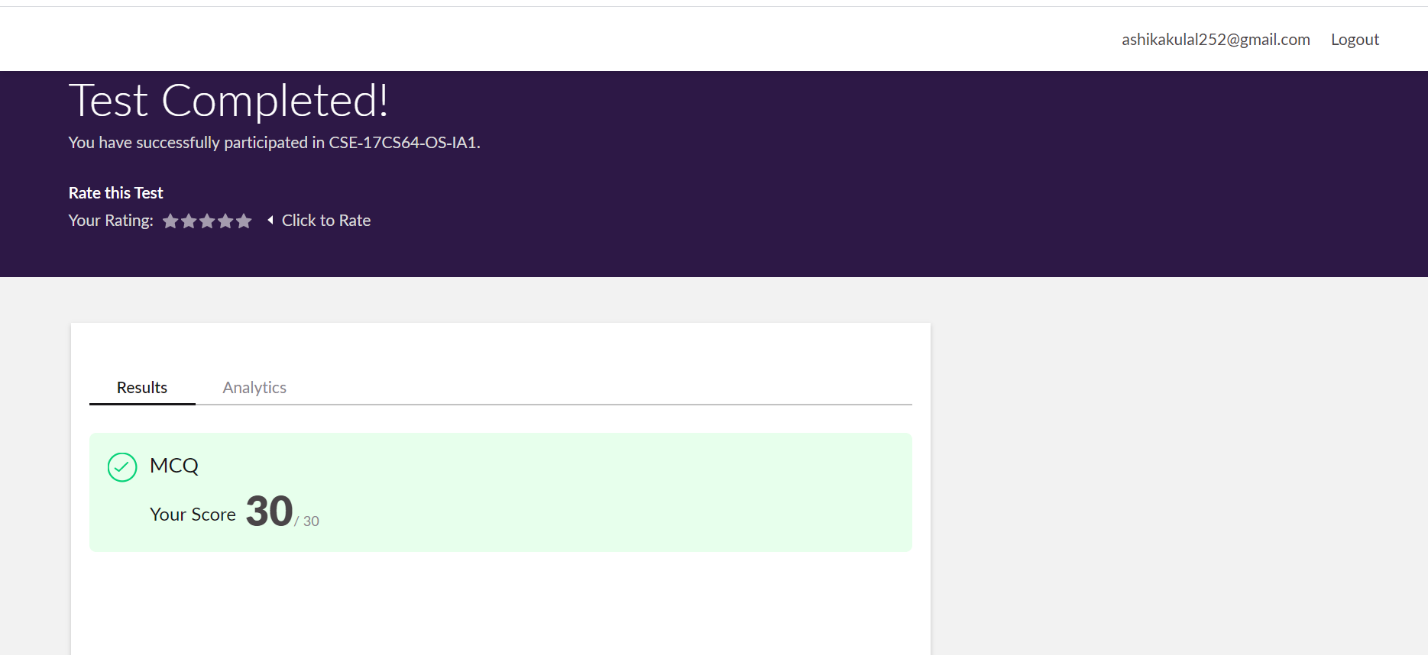
**DAILY ONLINE ACTIVITIES SUMMARY**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Date:** | **21-05-2020** | | | | | **Name:** | **ASHIKA** | |
| **Sem & Sec** | **A** | | | | | **USN:** | **4AL17CS016** | |
| **Online Test Summary** | | | | | | | | |
| **Subject** | | **Operating System.** | | | | | | |
| **Max. Marks** | | **30** | | **Score** | | | **30** | |
| **Certification Course Summary** | | | | | | | | |
| **Course** | **FRONT-END DEVELOPMENT OF HTML** | | | | | | | |
| **Certificate Provider** | | | Greatlearning  Academy | | **Duration** | | | 3.5HOURS |
| **Coding Challenges** | | | | | | | | |
| **Problem Statement:** 1.Getting a message printed through Applet.  2. AppletDemo  3.  Write C Program to create Singly Liked List with n elements and reverse the elements  4. Write a Java Program to Demonstrate a Basic Calculator using Applet  5.Write a C program to construct a singly linked list by removing duplicate elements in the sorted linked list  6. Write a java program to implement round robin scheduling algorithm.Calculate AVG WT AND TAT.  7.  write a simple applet java program to check whether the given number is armstrong number or not | | | | | | | | |
| **Status: Done** | | | | | | | | |
| **Uploaded the report in Github** | | | | | **YES** | | | |
| **If yes Repository name** | | | | | **Online report2** | | | |
| **Uploaded the report in slack** | | | | | **YES** | | | |

Online Test Details:

Subject:-Operating System.



Certification Course Details:

**Introduction to html:**

Today I have studied front-end development of html

Under this topic I have studied:-

\* Block level elements

\* inline level

\*inline vs blocks

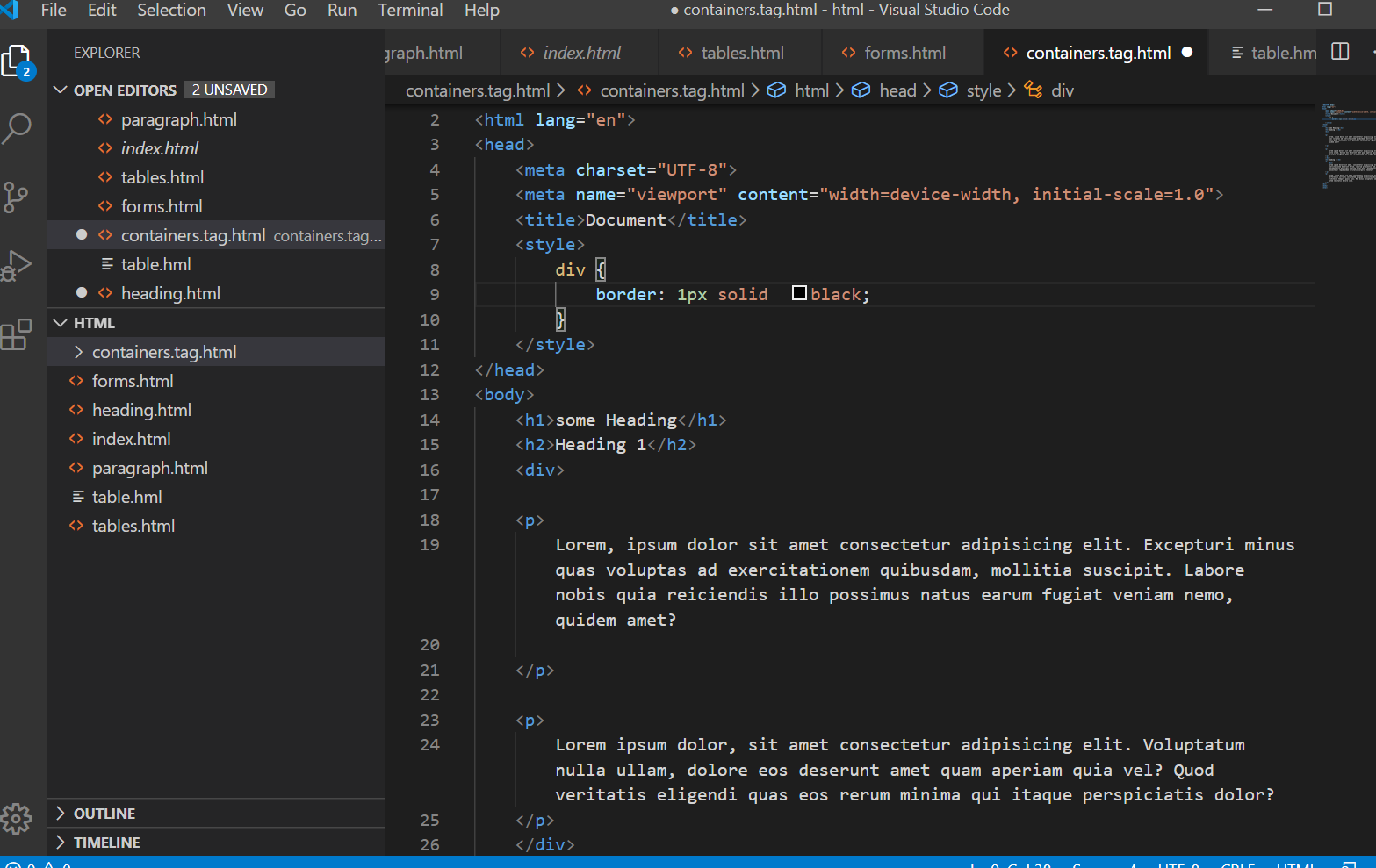
\* ID

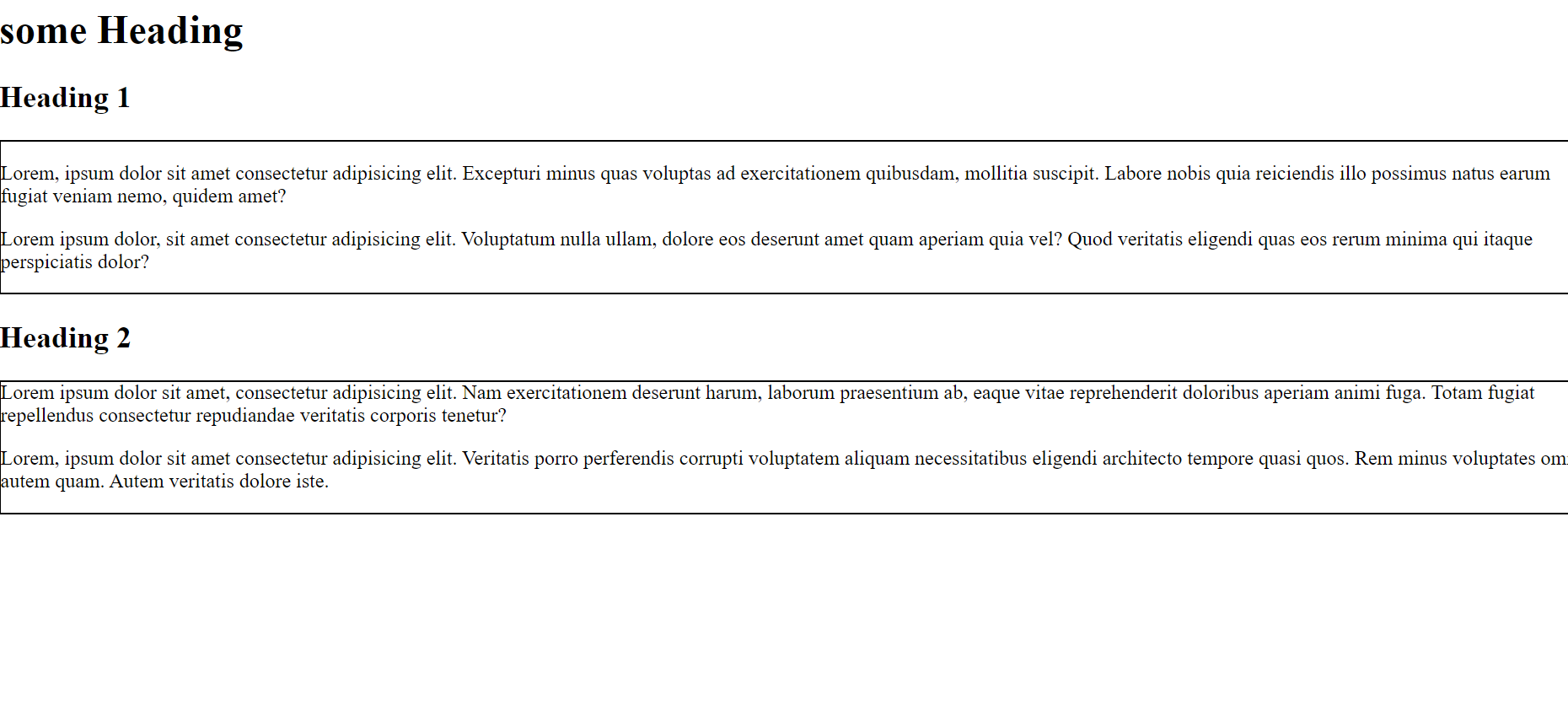
\* classes

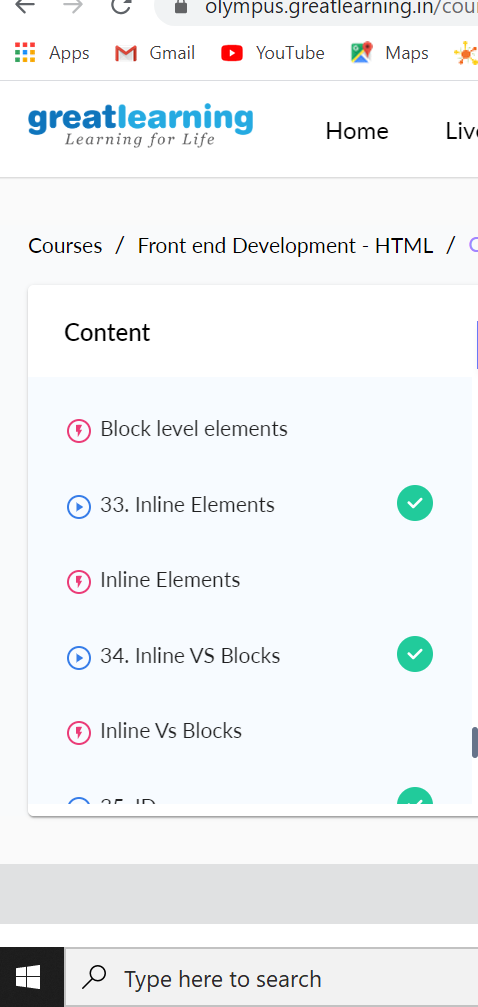
\* Relationship between elements

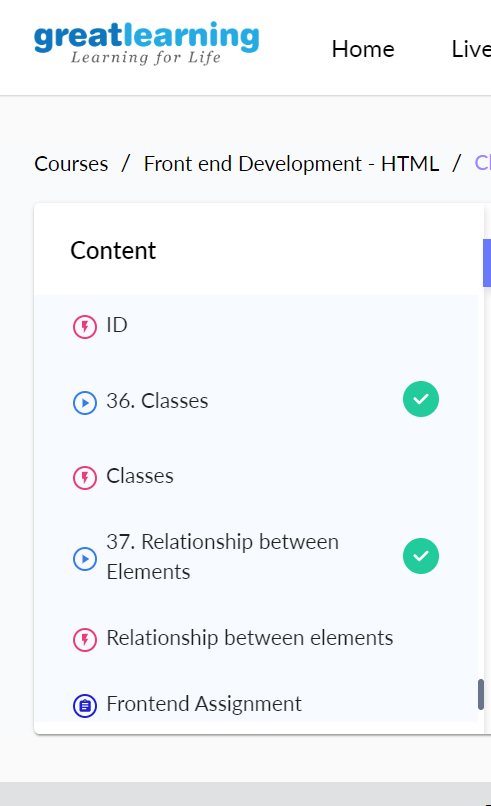
\* with quiz and programming assignments

.









Coding Challenges Details:

**1.Getting a message printed through Applet**

import java.awt.\*;  
import java.applet.\*;  
public class AppletDemo  
{  
      public void paint(Graphics g)  
      {  
           g.drawString("Welcome to TutorialRide", 50, 50);  
      }  
}

**//AppletDemo.html**  
  
<html>  
      <head>  
            <title> AppletDemo </title>  
      </head>  
      <body>  
            <applet code="AppletDemo.class" width="200" height="250">  
            </applet>  
      </body>  
</html>

**2.** **<title> AppletDemo </title>**

import java.applet.\*;

import java.awt.\*;

public class Simple extends Applet {

public void paint (Graphics g) {

g.drawString (" A Simple Applet", 20, 20);

}

}

**3.**  Write C Program to create Singly Liked List with n elements and reverse the elements

#include <stdio.h>

struct node{

int data;

struct node \*next;

};

struct node \*head, \*tail = NULL;

void addNode(int data) {

struct node \*newNode = (struct node\*)malloc(sizeof(struct node));

newNode->data = data;

newNode->next = NULL;

if(head == NULL) {

head = newNode;

tail = newNode;

}

else {

tail->next = newNode;

tail = newNode;

}

}

void reverse(struct node \*current) {

if(head == NULL) {

printf("List is empty\n");

return;

}

else{

if(current->next == NULL) {

printf("%d ", current->data);

return;

}

reverse(current->next);

printf("%d ", current->data);

}

}

void display() {

struct node \*current = head;

if(head == NULL) {

printf("List is empty\n");

return;

}

while(current != NULL) {

printf("%d ", current->data);

current = current->next;

}

printf("\n");

}

int main()

{

addNode(1);

addNode(2);

addNode(3);

addNode(4);

printf("Original List: \n");

display();

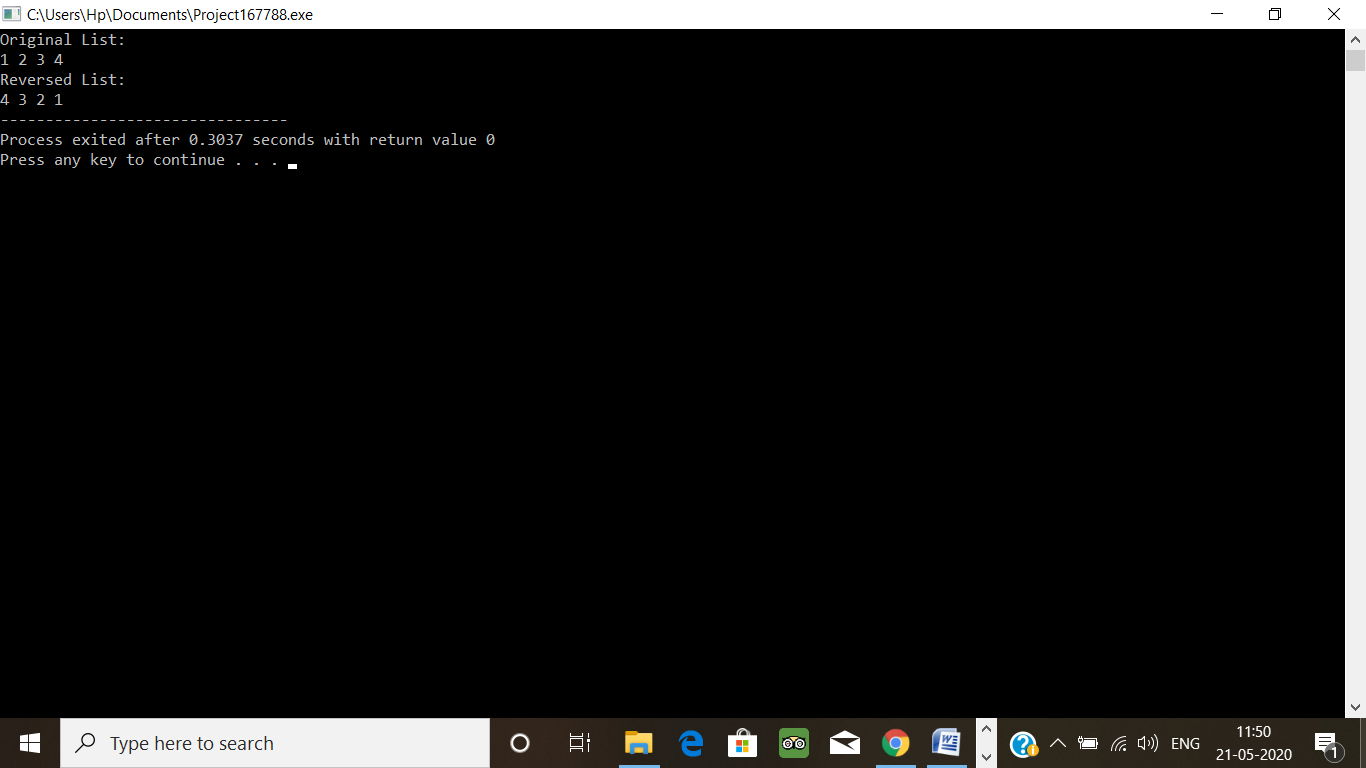
printf("Reversed List: \n");

reverse(head);

return 0;

}

**Output:**

****

**4.** Write a Java Program to Demonstrate a Basic Calculator using Applet

Problem Description  
We have to write a program in Java such that it creates a calculator which allows basic operations of addition, subtraction, multiplication and division.

Expected Input and Output  
For creating a calculator, we can have the following different sets of input and output.

To Perform Addition :  
When the addition expression "58+10" is typed,  
it is expected that the result is displayed as "58+10=68.0".  
2. To Perform Subtraction :

When the subtraction expression "100.0-28.25" is typed,  
it is expected that the result is displayed as "100.0-28.25=71.75".  
3. To Perform Multiplication :

When an multiplication expression "113.6539" is typed,it is expected that the result is displayed as "113.6539=4432.35".  
4. To Perform Division : When the denominator is non-zero

When an division expression "25126.0/3" is typed,  
it is expected that the result is displayed as "25126.0/3=8375.33".  
5. To Perform Division : When the denominator is zero

When an division expression "169.0/0" is typed,  
it is expected that the error is displayed as "169.0/0=Zero Divison Error".

import java.awt.\*;

import java.applet.\*;

import java.awt.event.\*;

public class Calculator extends Applet implements ActionListener

{

TextField inp;

public void init()

{

setBackground(Color.white);

setLayout(null);

int i;

inp = new TextField();

inp.setBounds(150,100,270,50);

this.add(inp);

Button button[] = new Button[10];

for(i=0;i<10;i++)

{

button[i] = new Button(String.valueOf(9-i));

button[i].setBounds(150+((i%3)\*50),150+((i/3)\*50),50,50);

this.add(button[i]);

button[i].addActionListener(this);

}

Button dec=new Button(".");

dec.setBounds(200,300,50,50);

this.add(dec);

dec.addActionListener(this);

Button clr=new Button("C");

clr.setBounds(250,300,50,50);

this.add(clr);

clr.addActionListener(this);

Button operator[] = new Button[5];

operator[0]=new Button("/");

operator[1]=new Button("\*");

operator[2]=new Button("-");

operator[3]=new Button("+");

operator[4]=new Button("=");

for(i=0;i<4;i++)

{

operator[i].setBounds(300,150+(i\*50),50,50);

this.add(operator[i]);

operator[i].addActionListener(this);

}

operator[4].setBounds(350,300,70,50);

this.add(operator[4]);

operator[4].addActionListener(this);

}

String num1="";

String op="";

String num2="";

//Function to calculate the expression

public void actionPerformed(ActionEvent e)

{

String button = e.getActionCommand();

char ch = button.charAt(0);

if(ch>='0' && ch<='9'|| ch=='.')

{

if (!op.equals(""))

num2 = num2 + button;

else

num1 = num1 + button;

inp.setText(num1+op+num2);

}

else if(ch=='C')

{

num1 = op = num2 = "";

inp.setText("");

}

else if (ch =='=')

{

if(!num1.equals("") && !num2.equals(""))

{

double temp;

double n1=Double.parseDouble(num1);

double n2=Double.parseDouble(num2);

if(n2==0 && op.equals("/"))

{

inp.setText(num1+op+num2+" = Zero Division Error");

num1 = op = num2 = "";

}

else

{

if (op.equals("+"))

temp = n1 + n2;

else if (op.equals("-"))

temp = n1 - n2;

else if (op.equals("/"))

temp = n1/n2;

else

temp = n1\*n2;

inp.setText(num1+op+num2+" = "+temp);

num1 = Double.toString(temp);

op = num2 = "";

}

}

else

{

num1 = op = num2 = "";

inp.setText("");

}

}

else

{

if (op.equals("") || num2.equals(""))

op = button;

else

{

double temp;

double n1=Double.parseDouble(num1);

double n2=Double.parseDouble(num2);

if(n2==0 && op.equals("/"))

{

inp.setText(num1+op+num2+" = Zero Division Error");

num1 = op = num2 = "";

}

else

{

if (op.equals("+"))

temp = n1 + n2;

else if (op.equals("-"))

temp = n1 - n2;

else if (op.equals("/"))

temp = n1/n2;

else

temp = n1\*n2;

num1 = Double.toString(temp);

op = button;

num2 = "";

}

}

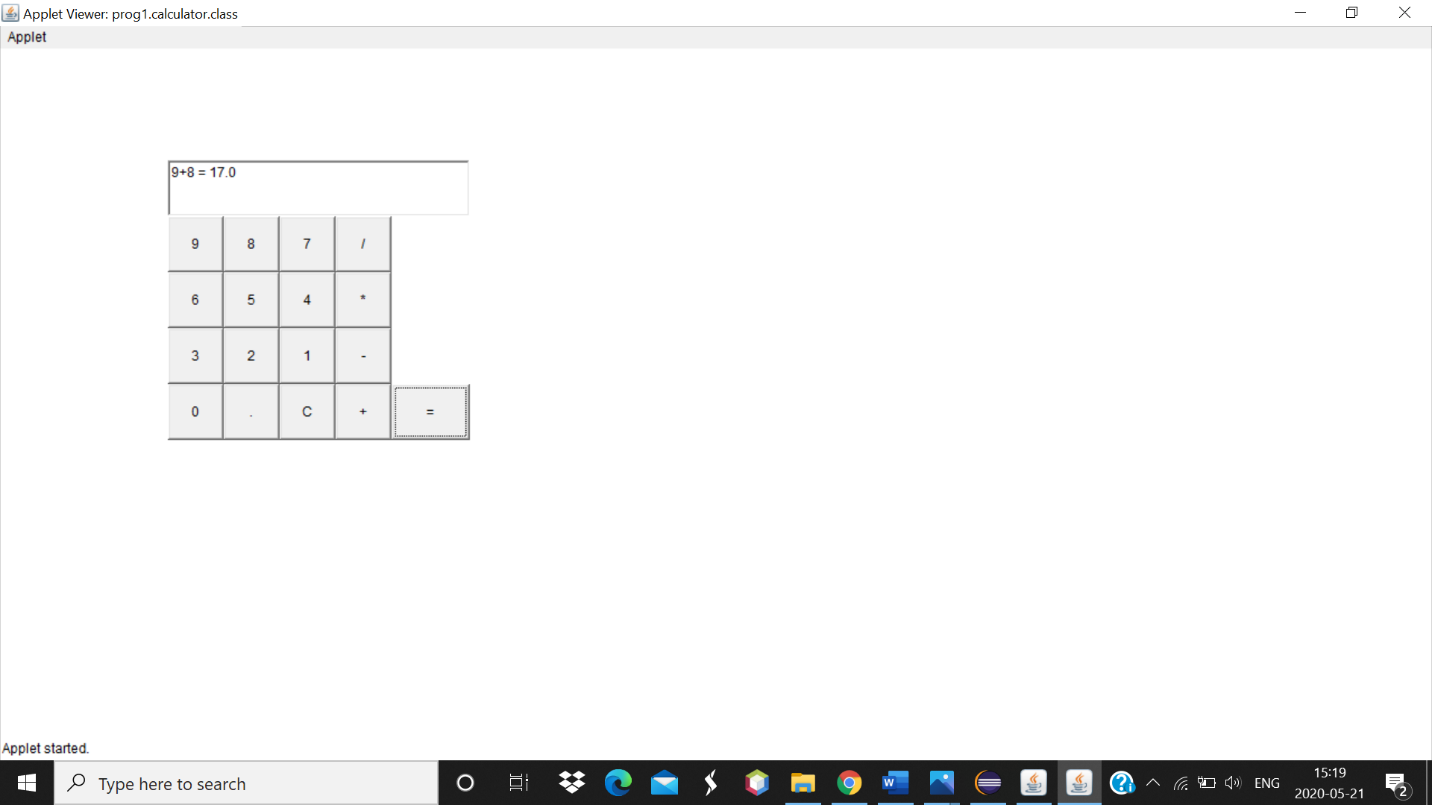
inp.setText(num1+op+num2);

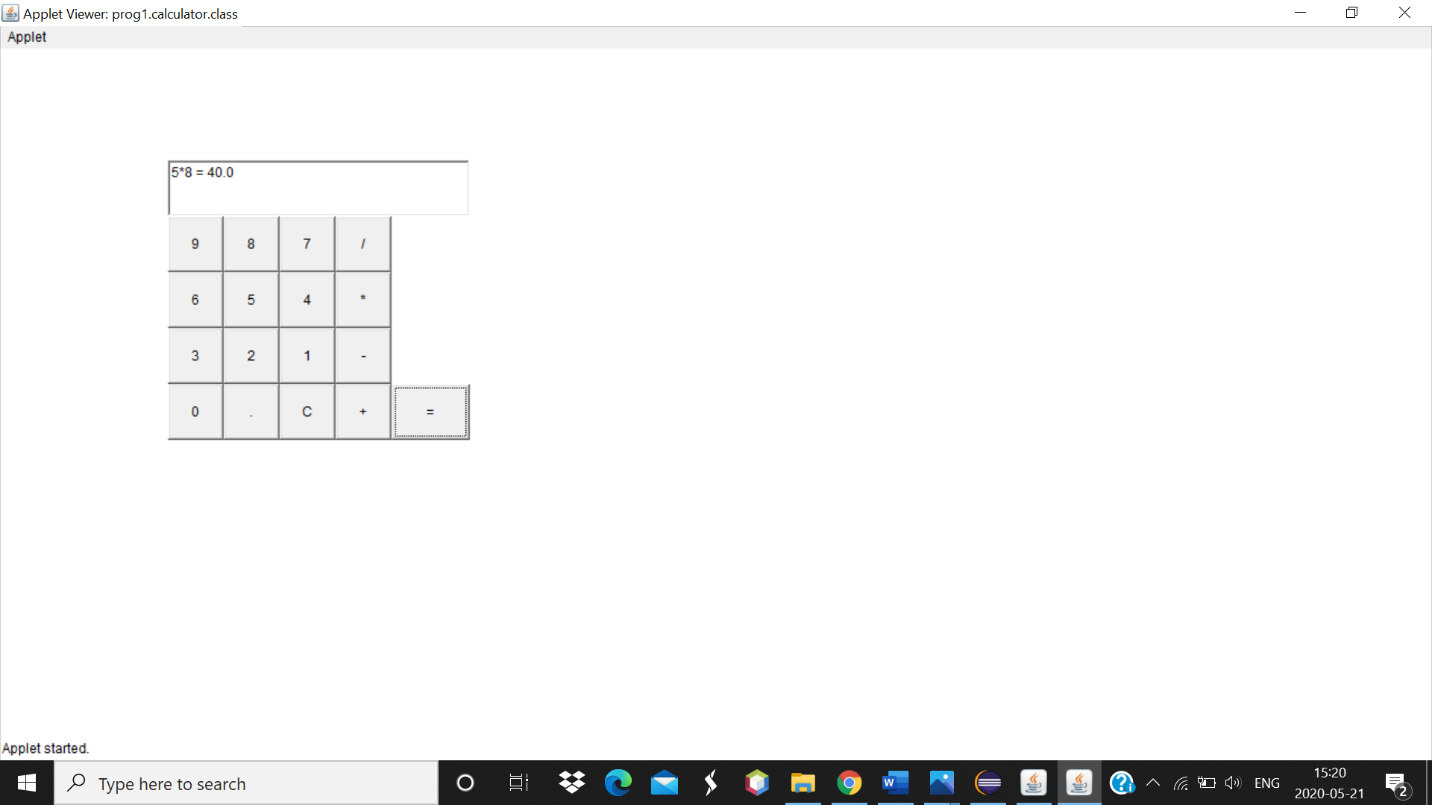
}

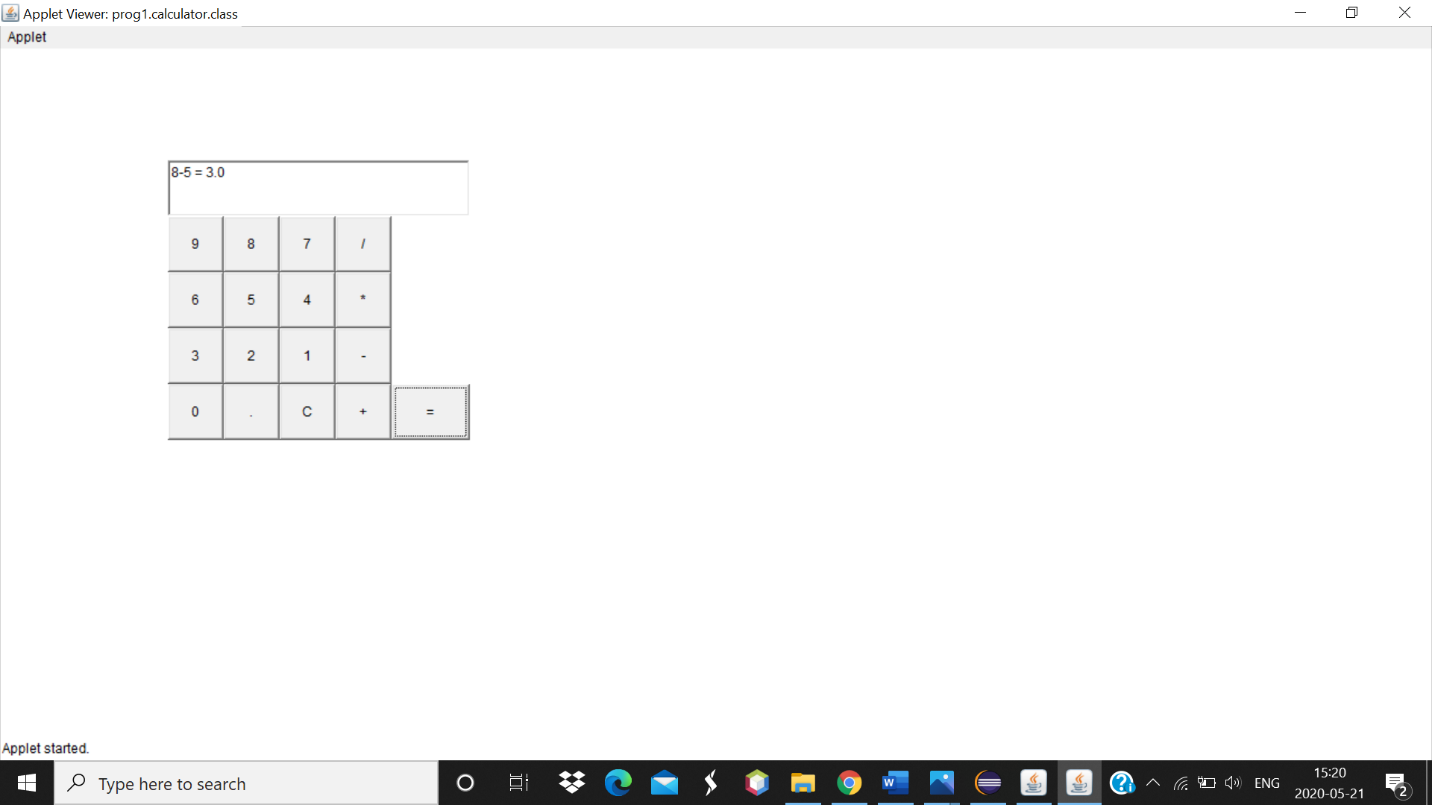
}

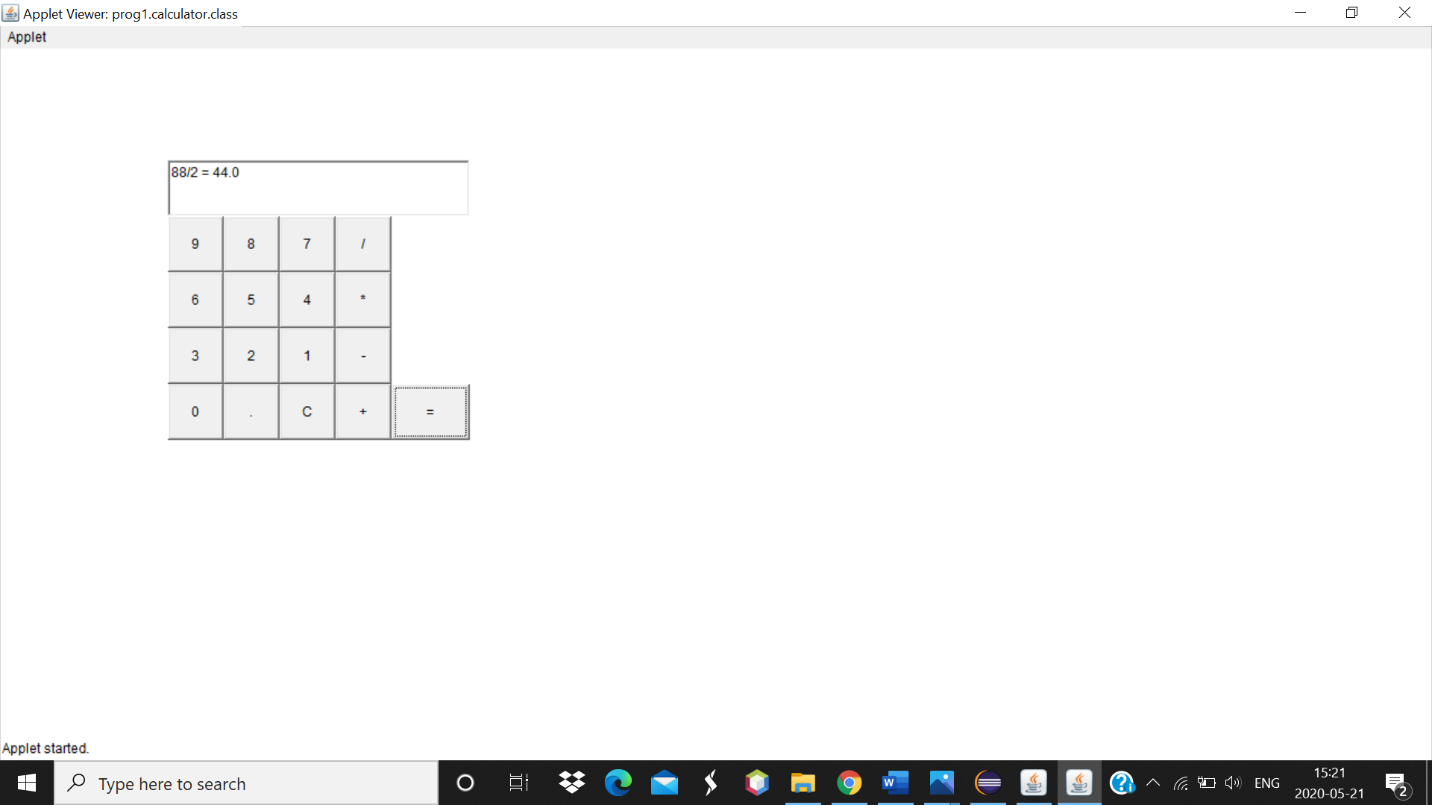
}

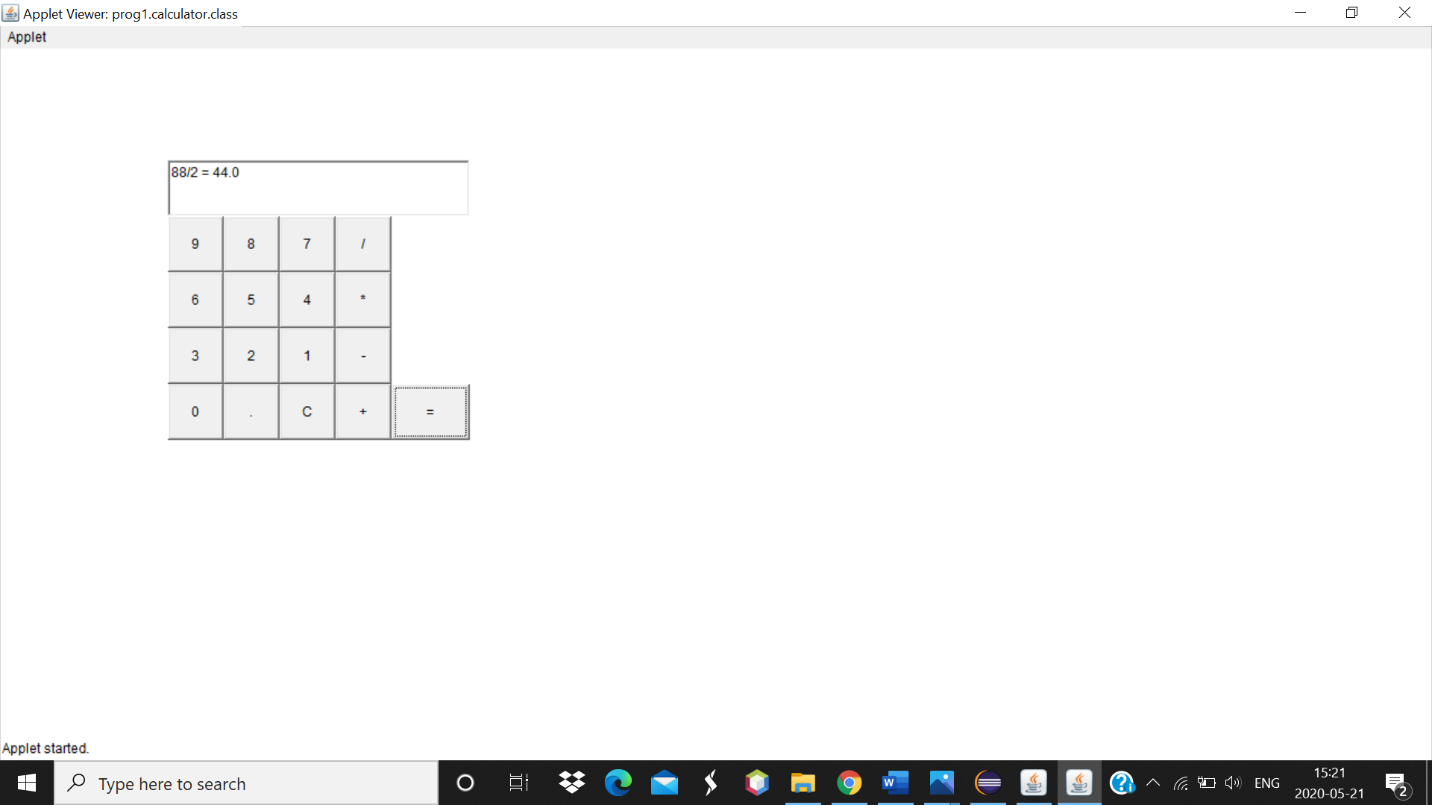
**Output:**











5.Write a C program to construct a singly linked list by removing duplicate elements in the sorted linked list  
Description:  
Take a sorted list and traverse the list. Compare the current node element with next adjacent node. If it is same then delete second element, if not retain. Finally print the resulting list.  
Sample output:  
Given list {1,2,2,3,3,3,4}  
Resulting list{1,2,3,4}

#include <stdio.h>

struct node{

int data;

struct node \*next;

};

struct node \*head, \*tail = NULL;

void addNode(int data) {

struct node \*newNode = (struct node\*)malloc(sizeof(struct node));

newNode->data = data;

newNode->next = NULL;

if(head == NULL) {

head = newNode;

tail = newNode;

}

else {

tail->next = newNode;

tail = newNode;

}

}

void removeDuplicate() {

struct node \*current = head, \*index = NULL, \*temp = NULL;

if(head == NULL) {

return;

}

else {

while(current != NULL){

temp = current;

index = current->next;

while(index != NULL) {

if(current->data == index->data)

{

temp->next = index->next;

}

else {

temp = index;

}

index = index->next;

}

current = current->next;

}

}

}

void display() {

struct node \*current = head;

if(head == NULL) {

printf("List is empty \n");

return;

}

while(current != NULL) {

printf("%d ", current->data);

current = current->next;

}

printf("\n");

}

int main()

{

addNode(1);

addNode(2);

addNode(3);

addNode(2);

addNode(2);

addNode(4);

addNode(1);

printf("Originals list: \n");

display();

removeDuplicate();

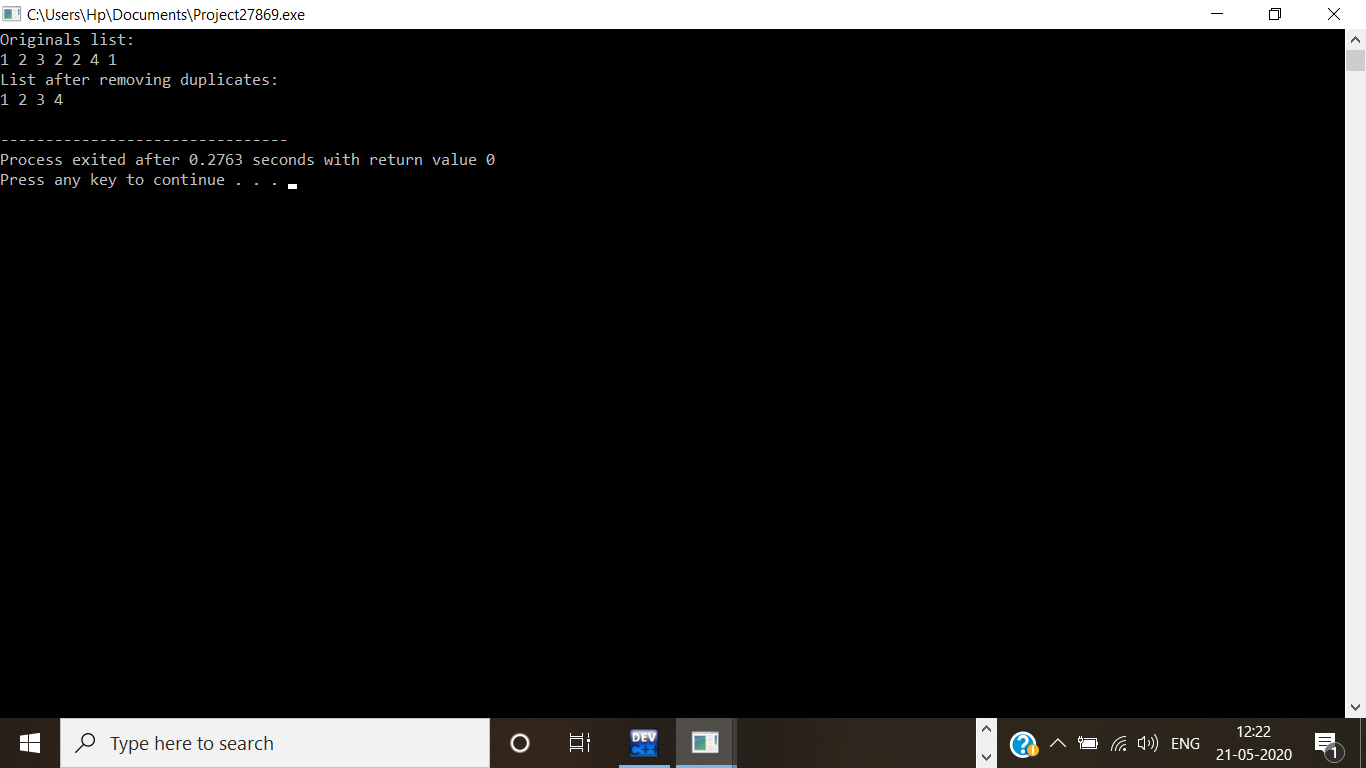
printf("List after removing duplicates: \n");

display();

return 0;

}

**Ouptput:**



**6.** Write a java program to implement round robin scheduling algorithm.Calculate AVG WT AND TAT.

public class Main

{

static void findWaitingTime(int processes[], int n,

int bt[], int wt[], int quantum)

{

int rem\_bt[] = new int[n];

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

rem\_bt[i] = bt[i];

int t = 0;

while(true)

{

boolean done = true;

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

{

if (rem\_bt[i] > 0)

{

done = false;

if (rem\_bt[i] > quantum)

{

t += quantum;

rem\_bt[i] -= quantum;

}

else

{

t = t + rem\_bt[i];

wt[i] = t - bt[i];

rem\_bt[i] = 0;

}

}

}

if (done == true)

break;

}

}

static void findTurnAroundTime(int processes[], int n,

int bt[], int wt[], int tat[])

{

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

tat[i] = bt[i] + wt[i];

}

static void findavgTime(int processes[], int n, int bt[],

int quantum)

{

int wt[] = new int[n], tat[] = new int[n];

int total\_wt = 0, total\_tat=0;

findWaitingTime(processes, n, bt, wt, quantum);

findTurnAroundTime(processes, n, bt, wt, tat);

System.out.println("Processes " + " Burst time " +

" Waiting time " + " Turn around time");

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

{

total\_wt = total\_wt + wt[i];

total\_tat = total\_tat + tat[i];

System.out.println(" " + (i+1) + "\t\t" + bt[i] +"\t " +

wt[i] +"\t\t " + tat[i]);

}

System.out.println("Average waiting time = " +

(float)total\_wt / (float)n);

System.out.println("Average turn around time = " +

(float)total\_tat / (float)n);

}

public static void main(String[] args)

{

int processes[] = { 1, 2, 3};

int n = processes.length;

int burst\_time[] = {10, 5, 8};

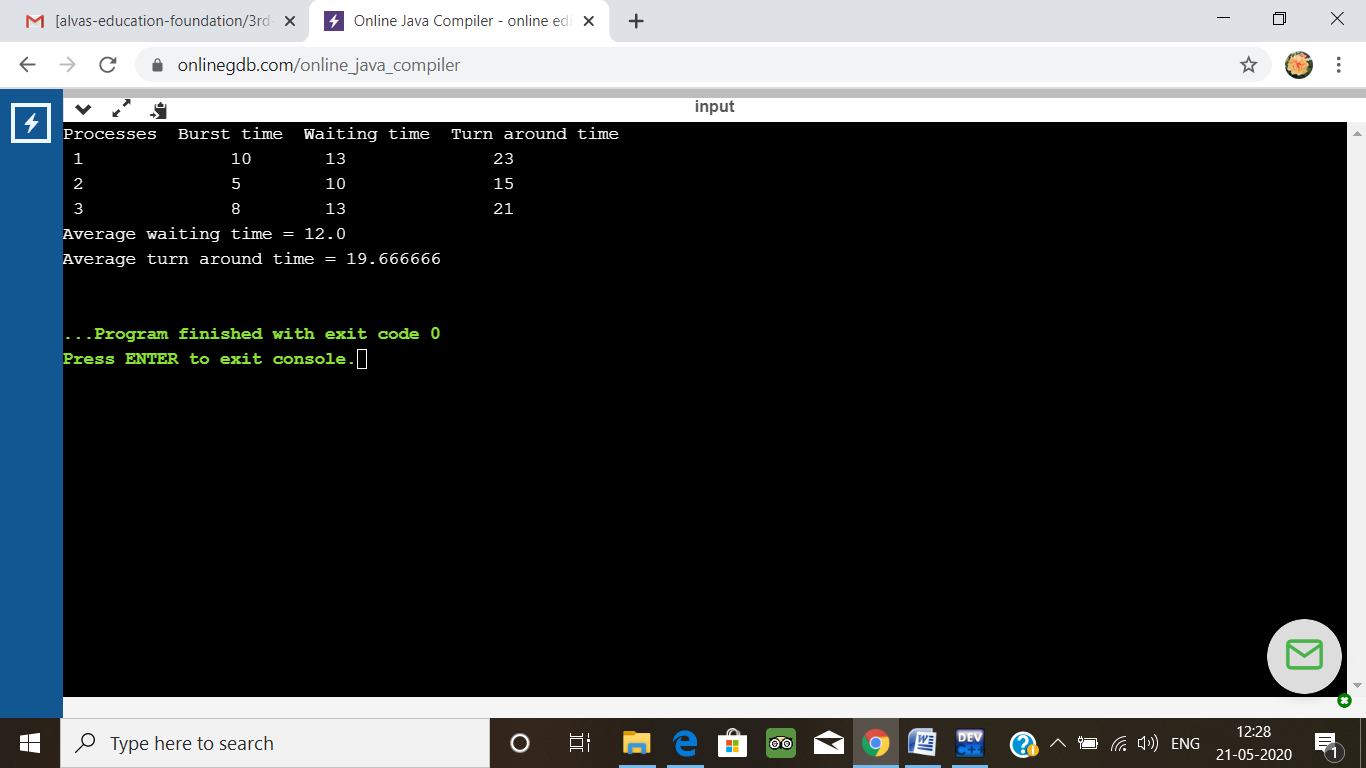
int quantum = 2;

findavgTime(processes, n, burst\_time, quantum);

}

}

**OUTPUT:**

****

**7.**  write a simple applet java program to check whether the given number is armstrong number or not

**Description**  
Armstrong Number :A positive number is called armstrong number if it is equal to the sum of cubes of its digits for example 0, 1, 153, 370, 371, 407 etc.

Let's try to understand why 153 is an Armstrong number.  
153 = (111)+(555)+(333)  
where:  
(111)=1  
(555)=125  
(333)=27  
So:  
1+125+27=153

import java.util.Scanner;

public class Astro {

public static void main(String[] args) {

int num, number, temp, total = 0;

System.out.println("Ënter 3 Digit Number");

Scanner scanner = new Scanner(System.in);

num = scanner.nextInt();

scanner.close();

number = num;

for( ;number!=0;number /= 10)

{

temp = number % 10;

total = total + temp\*temp\*temp;

}

if(total == num)

System.out.println(num + " is an Armstrong number");

else

System.out.println(num + " is not an Armstrong number");

}

}

**Output:**

