**LAB 3 – IN LAB**

**Question 1:**Source Code:

#include <iostream>

using namespace std;

class Time{

    int hours;

    int minutes;

    int seconds;

    public:

    Time(){

        this->hours = 0;

        this->minutes = 0;

        this->seconds = 0;

    }

    Time(int hours, int minutes, int seconds){

        this->hours = hours;

        this->minutes = minutes;

        this->seconds = seconds;

    }

    void DisplayTime(Time t3){

        cout << "Time: " << t3.hours << ":" << t3.minutes << ":" << t3.seconds << endl;

        return;

    }

    Time AddTime(Time t2){

        Time t3;

         t3.seconds =   seconds + t2.seconds;

         t3.minutes =   minutes + t2.minutes;

         t3.hours =   hours + t2.hours;

         return t3;

    }

};

int main(){

    Time();

    Time O1(4,50,35);

    Time O2(5,20,3);

    Time O3;

    O3 = O1.AddTime(O2);

    O3.DisplayTime(O3);

}

Output:  


**Question 2:**Source Code:

#include<iostream>

#include<conio.h>

using namespace std;

class toolBooth{

    public:

    unsigned int totalCars;

    double MoneyCollected;

    public:

    toolBooth(){

        totalCars =0;

        MoneyCollected =0;

    }

    void payingCar(){

        totalCars++;

        MoneyCollected+=0.5;

    }

    void noPayCar(){

        totalCars++;

    }

    void Display(){

        cout<<"\n Total Cars passed : "<<totalCars<<endl;

        cout<<"Total Money Collected : "<<MoneyCollected<<endl;

    }

};

int main(){

    toolBooth T1;

    char input;

    while(1){

cout<<"Enter \np :Paying \nn : nonPaying"<<endl;

input = getch();

switch(input){

    case 'p':

    T1.payingCar();

    break;

    case 'n':

    T1.noPayCar();

    break;

    case 27:

    T1.Display();

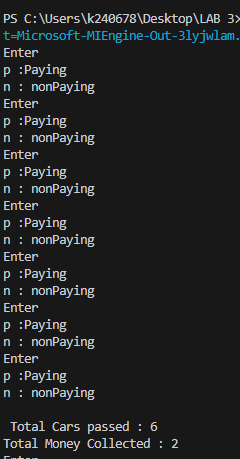
    break;

}

return 0;

    }

}

Output:  


**Question 3:**Source Code:

#include<iostream>

using namespace std;

class Serial{

    private:

    int serial;

    static int count;

    public:

    void Getter(){

        cout<<"Count is "<<count;

    }

    void SerialGenerator(){

        count++;

        serial = 100 + count;

    }

    void display(){

        SerialGenerator();

        cout<<" I am Object "<<serial<<endl;

    }

    };

    int Serial::count =0;

int main(){

    Serial S1,S2,S3;

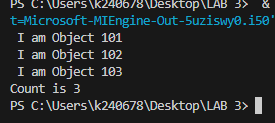
    S1.display();

    S2.display();

    S3.display();

    S1.Getter();

}

Output:  


**Question 4:**Source Code:

#include<iostream>

#include<iomanip>

using namespace std;

class Location{

    int degree;

    float minutes;

    char DirectionalLetter;

public:

    Location(int degree, float minutes, char DirectionalLetter){

        this->degree = degree;

        this->minutes = minutes;

        this->DirectionalLetter = DirectionalLetter;

    }

    void Validate(int degree,float minutes,char DirectionalLetter){

        if(degree<0 || degree>180){

            cout<<"Invalid Degree. Degree should be between 0 and 90."<<endl;

            exit(0);

        }

        if(minutes<0 || minutes>60){

            cout<<"Invalid Minutes. Minutes should be between 0 and 60."<<endl;

            exit(0);

        }

        if(DirectionalLetter!='N' && DirectionalLetter!='S' && DirectionalLetter!='E' && DirectionalLetter!='W' &&DirectionalLetter!='n' && DirectionalLetter!='s' && DirectionalLetter!='e' && DirectionalLetter!='w'){

            cout<<"Invalid Directional Letter. Directional Letter should be either N/S/E/W."<<endl;

            exit(0);

        }

    }

    void InputVariables(){

        cout<<"Enter Degree: ";

        cin>>degree;

        cout<<"Enter Minutes: ";

        cin>>minutes;

        cout<<"Enter Directional Letter (N/S/E/W): ";

        cin>>DirectionalLetter;

        Validate(degree, minutes, DirectionalLetter);

    }

    void OutputLocation(){

        cout<<"Location : "<<degree<<"\xF8,"<<minutes<<"'"<<DirectionalLetter<<endl;

    }

};

int main(){

    while(1){

    Location location(150,60.5,'E');

    location.InputVariables();

    location.OutputLocation();

    }

}

Output:  
