**Task 1 : 30 marks**

1. Create a Time class to store hours and minutes. Implement:
   1. Overload the + operator to add two Time objects
   2. Overload the > operator to compare two Time objects
   3. Handle invalid time (>24 hours or >60 minutes) by throwing a custom exception

SOLUTION:

#include <iostream>

using namespace std;

class Time

{

private:

int hrs , mins;

public:

Time ()

{

hrs=0;

mins=0;

}

void sett(int h , int m)

{

if (h>=24 || h<0 || m<0 || m>=60)

{

cout << "INVALID TIME "<<endl;

}

else

{

hrs= h;

mins= m;

}

}

Time operator+(Time t)

{

Time res;

int totalmin = mins + t.mins;

int totalhrs = hrs + t.hrs + (totalmin / 60 );

res.hrs = totalhrs % 24;

res.mins = totalmin % 60;

return res;

}

bool operator>(Time t)

{

if (hrs > t.hrs)

return true;

else if (hrs == t.hrs && mins > t.mins)

return true;

else

return false;

}

void display ()

{

cout<< hrs << "hours and "<<mins<< " minutes "<<endl;

}

};

int main()

{

Time t1, t2 , result;

t1.sett(10,30);

t2.sett(3,46);

result = t1+t2;

cout<<"RESULT OF ADDING ";

result.display();

if (t1 > t2) {

cout << "t1 is greater than t2" << endl;

} else {

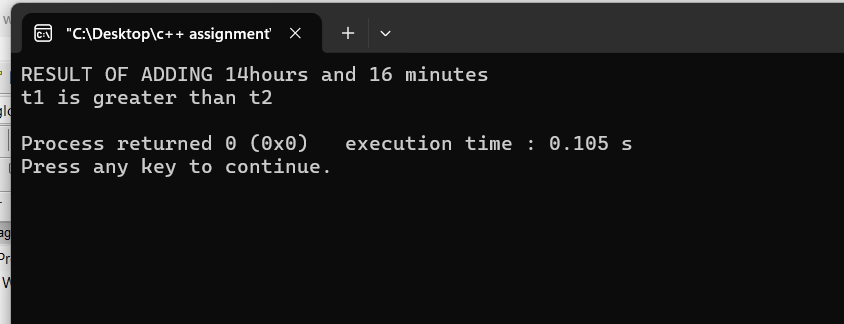
cout << "t2 is greater than t1" << endl;

}

return 0;

}

OUTPUT:



**Task 2: 70 marks**

1. Create a base class Vehicle and two derived classes Car and Bike:
   1. Vehicle has registration number and color
   2. Car adds number of seats
   3. Bike adds engine capacity
   4. Each class should have its own method to write its details to a file
   5. Include proper inheritance and method overriding

SOLUTION:

#include <iostream>

#include <fstream>

using namespace std;

class Vehicle

{

public:

string regnum ;

string color ;

virtual void input() //usual input

{

cout << " Enter registration number: ";

cin >> regnum;

cout << " Enter color: ";

cin>> color;

}

virtual ~Vehicle() {}

void insertFile()

{

ofstream file("vehicle.txt");

file<<" Registration: "<<regnum << "Color: "<<color<<endl;

file.close();

}

};

class Car : public Vehicle

{

public:

int seats;

void input() override

{

Vehicle::input(); //takes input as usual

cout << "Enter number of seats: ";

cin >> seats;

}

void writecar()

{

ofstream file("car.txt");

file<<" Registration: "<<regnum << "Color: "<<color<< "Seats: "<<seats <<endl;

file.close();

}

void displaycar()

{

cout << "\n--- Car Details ---\n";

cout << "Registration: " << regnum << endl;

cout << "Color: " << color << endl;

cout << "Seats: " << seats << endl;

}

};

class Bike: public Vehicle

{

public:

int ecap;

void input() override

{

Vehicle::input();

cout<< "Enter engine capacity "<<endl;

cin>>ecap;

}

void writebike()

{

ofstream file("bike.txt");

file << "Registration: "<<regnum << "Color: "<<color<< "Engine Capacity:"<<ecap<<endl;

file.close();

}

void displaybike()

{

cout << "\n--- Bike Details ---\n";

cout << "Registration: " << regnum << endl;

cout << "Color: " << color << endl;

cout << "Engine Capacity: " << ecap << endl;

}

};

int main()

{

Car c ;

Bike b;

cout<<" FOR CAR "<<endl;

c.input();

c.writecar();

c.displaycar();

cout<<" FOR BIKE "<<endl;

b.input();

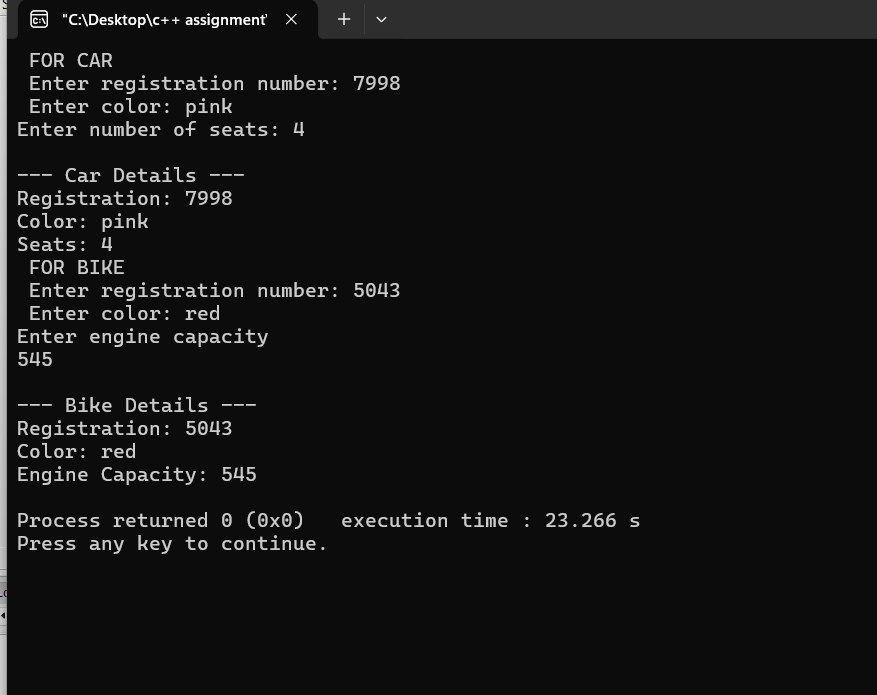
b.writebike();

b.displaybike();

return 0;

}

OUTPUT:



1. Create a program that:
   1. Reads student records (roll, name, marks) from a text file
   2. Throws an exception if marks are not between 0 and 100
   3. Allows adding new records with proper validation
   4. Saves modified records back to file

SOLUTION:

**#include <iostream>**

**#include <fstream>**

**#include <string>**

**using namespace std;**

**class student**

**{**

**private:**

**float mark1,mark2,mark3,roll;**

**string name;**

**char mod;**

**char grade;**

**public:**

**void addinput()**

**{**

**ofstream file("studentrecord.txt",ios::app);**

**if (file.is\_open())**

**{**

**cout<<"Enter your full name "<<endl;**

**getline(cin,name);**

**file<<"Name : "<<name<<endl;**

**cout<<"Enter your roll number "<<endl;**

**cin>>roll;**

**file<<"Roll no : "<<roll<<endl;**

**cin.ignore();**

**cout<<"Enter marks of Subject 1 out of 100 "<<endl;**

**cin>>mark1;**

**if (mark1>=0 && mark1<=100)**

**{**

**file<<"Mark1 : "<<mark1<<endl;**

**}**

**else**

**{**

**file << "Mark1 : INVALID MARK" << endl;**

**cout<<"INVALID MARK";**

**}**

**cout<<"Enter marks of Subject 2 out of 100 "<<endl;**

**cin>>mark2;**

**if (mark2>=0 && mark2<=100)**

**{**

**file<<"Mark 2 : "<<mark2<<endl;**

**}**

**else**

**{**

**file << "Mark2 : INVALID MARK" << endl;**

**cout<<"INVALID MARK";**

**}**

**cout<<"Enter marks of Subject 3 out of 100"<<endl;**

**cin>>mark3;**

**if (mark3>=0 && mark3<=100)**

**{**

**file<<"Mark 3 : "<<mark3<<endl;**

**}**

**else**

**{**

**file << "Mark3 : INVALID MARK" << endl;**

**cout<<"INVALID MARKS";**

**}**

**}**

**else**

**{**

**cout<<"File not open";**

**}**

**}**

**void modify()**

**{**

**cout <<"DO YOU WISH TO MODIFY RECORDS (y/n)"<<endl;**

**cin>>mod;**

**cin.ignore();**

**if (mod=='Y' || mod=='y')**

**{**

**string searchh;**

**string line;**

**cout<<"enter the name of the student you want to check: ";**

**getline(cin,searchh);**

**ifstream infile("studentrecord.txt");**

**ofstream temp("temp.txt");**

**if (!infile.is\_open()) {**

**cout << "File cannot be opened!" << endl;**

**return;**

**}**

**bool found = false;**

**while (getline(infile, line))**

**{**

**if (line.find("Name : " + searchh) != string::npos)**

**{**

**found=true;**

**temp << line << endl;**

**getline(infile, line); temp << line << endl;**

**cout << "Enter new marks for " << searchh << ":\n";**

**cout << "Subject 1: ";**

**cin >> mark1;**

**if (mark1 >= 0 && mark1 <= 100)**

**temp << "Mark1 : " << mark1 << endl;**

**else**

**temp << "Mark1 : INVALID MARK" << endl;**

**cout << "Subject 2: ";**

**cin >> mark2;**

**if (mark2 >= 0 && mark2 <= 100)**

**temp << "Mark2 : " << mark2 << endl;**

**else**

**temp << "Mark2 : INVALID MARK" << endl;**

**cout << "Subject 3: ";**

**cin >> mark3;**

**if (mark3>= 0 && mark3 <= 100)**

**temp << "Mark3 : " << mark3 << endl;**

**else**

**temp << "Mark3 : INVALID MARK" << endl;**

**temp << "Mark1 : " << mark1 << endl;**

**temp << "Mark2 : " << mark2 << endl;**

**temp << "Mark3 : " << mark3 << endl;**

**getline(infile, line);**

**getline(infile, line);**

**getline(infile, line);**

**}**

**else**

**{**

**temp<<line<<endl;**

**}**

**}**

**if (!found) {**

**cout << "Student not found!" << endl;**

**}**

**infile.close();**

**temp.close();**

**remove("studentrecord.txt");**

**rename("temp.txt", "studentrecord.txt");**

**}**

**else {**

**cout<<"END"<<endl;**

**}**

**}**

**void display()**

**{**

**ifstream openfile("studentrecord.txt");**

**string lines;**

**cout << "records : "<<endl;**

**while (getline(openfile, lines)) {**

**cout << lines << endl;**

**}**

**openfile.close();**

**}**

**};**

**int main(){**

**student s;**

**s.addinput();**

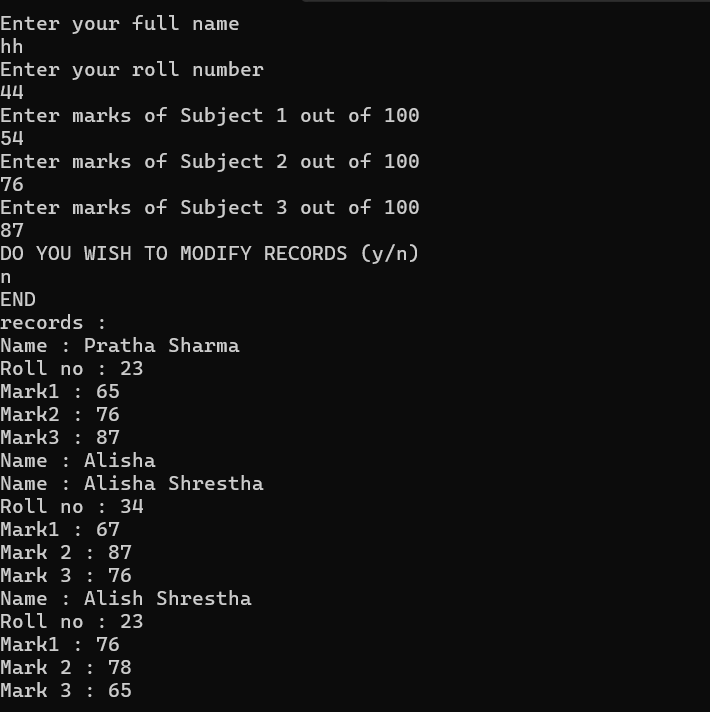
**s.modify();**

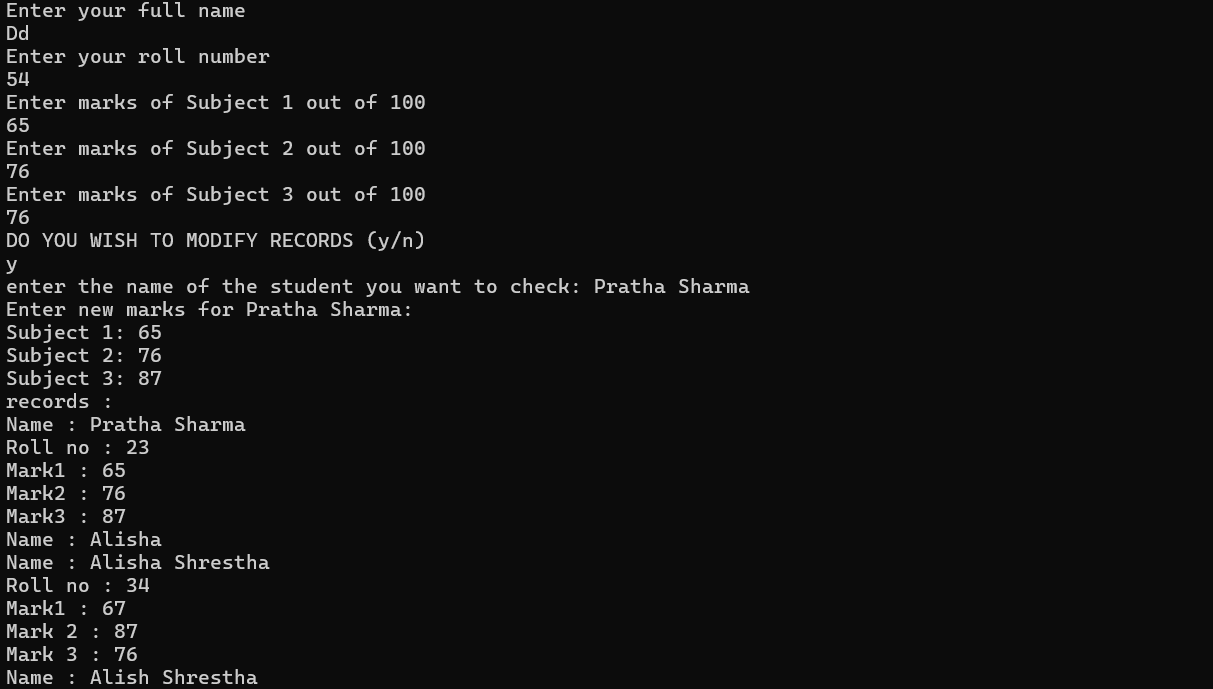
**s.display();**

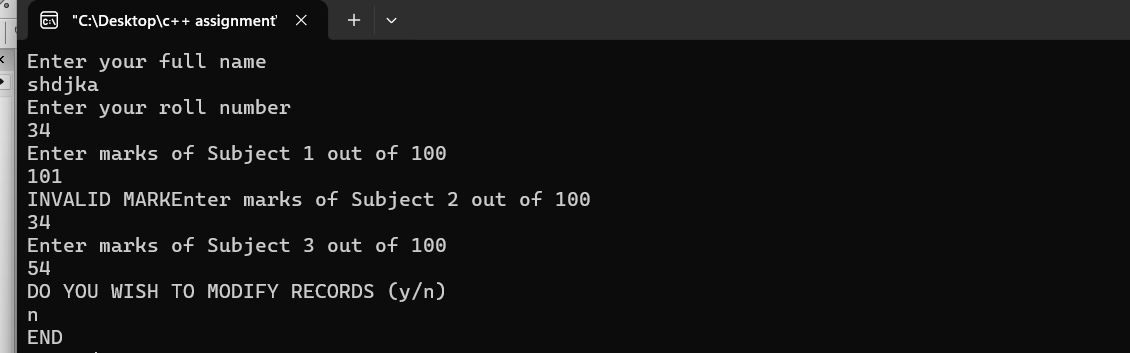
**return 0;**

**}**

OUTPUT:

****

****

****

1. **Task 3**
2. Check and commit all your solutions.