ASSIGNMNET

QUESTION # 1:

**Create a class Student with data members: name, roll\_number, and marks. • Implement member functions to input and display student data. • Create a program to: o Create a file named students.txt. o Write data of 5 students to the file using ofstream. o Read the data from the file and display it using ifstream. o Modify the marks of a specific student using fstream. C++**

CODE:

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

class Student {

public:

string name;

int roll\_number;

int marks;

void input() {

cout << "Enter name: ";

cin >> name;

cout << "Enter roll number: ";

cin >> roll\_number;

cout << "Enter marks: ";

cin >> marks;

}

void display() {

cout << "Name: " << name << endl;

cout << "Roll Number: " << roll\_number << endl;

cout << "Marks: " << marks << endl;

}

};

int main() {

Student students[5];

for (int i = 0; i < 5; i++) {

cout << "Enter data for student " << i + 1 << endl;

students[i].input();

}

ofstream file("students.txt");

if (file.is\_open()) {

for (int i = 0; i < 5; i++) {

file << students[i].name << endl;

file << students[i].roll\_number << endl;

file << students[i].marks << endl;

}

file.close();

cout << "Data written to file successfully." << endl;

} else {

cout << "Unable to open file for writing." << endl;

}

ifstream fileRead("students.txt");

if (fileRead.is\_open()) {

string name;

int roll\_number, marks;

for (int i = 0; i < 5; i++) {

getline(fileRead, name);

fileRead >> roll\_number >> marks;

fileRead.ignore();

cout << "Student " << i + 1 << ":" << endl;

cout << "Name: " << name << endl;

cout << "Roll Number: " << roll\_number << endl;

cout << "Marks: " << marks << endl;

cout << endl;

}

fileRead.close();

cout << "Data read from file successfully." << endl;

} else {

cout << "Unable to open file for reading." << endl;

}

int roll\_number\_to\_modify;

cout << "Enter roll number of student to modify marks: ";

cin >> roll\_number\_to\_modify;

fstream fileModify("students.txt", ios::in | ios::out);

if (fileModify.is\_open()) {

string name;

int roll\_number, marks;

int i = 0;

while (fileModify >> name >> roll\_number >> marks) {

if (roll\_number == roll\_number\_to\_modify) {

cout << "Enter new marks: ";

int new\_marks;

cin >> new\_marks;

fileModify.seekp(fileModify.tellg() - sizeof(marks));

fileModify.write((char\*)&new\_marks, sizeof(new\_marks));

cout << "Marks modified successfully." << endl;

break;

}

i++;

}

fileModify.close();

} else {

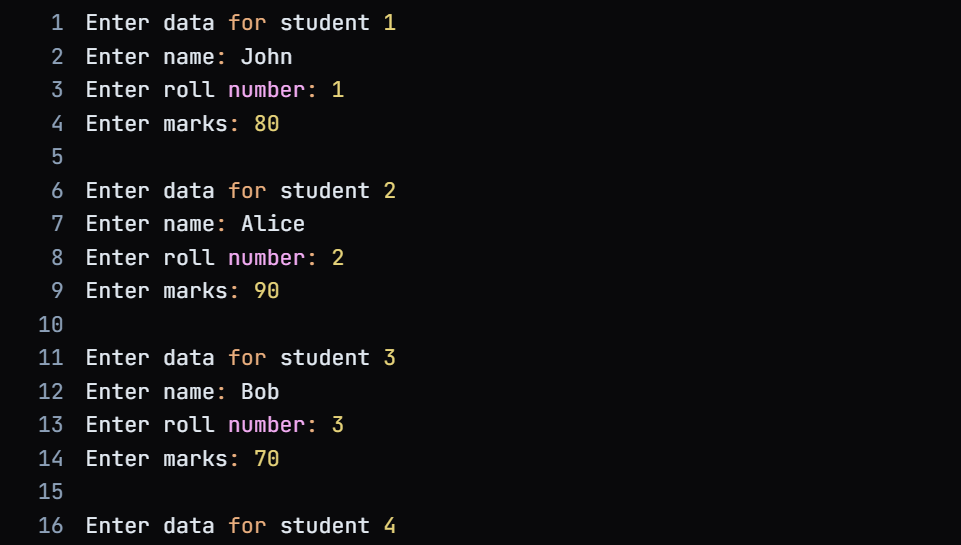
cout << "Unable to open file for modification." << endl;

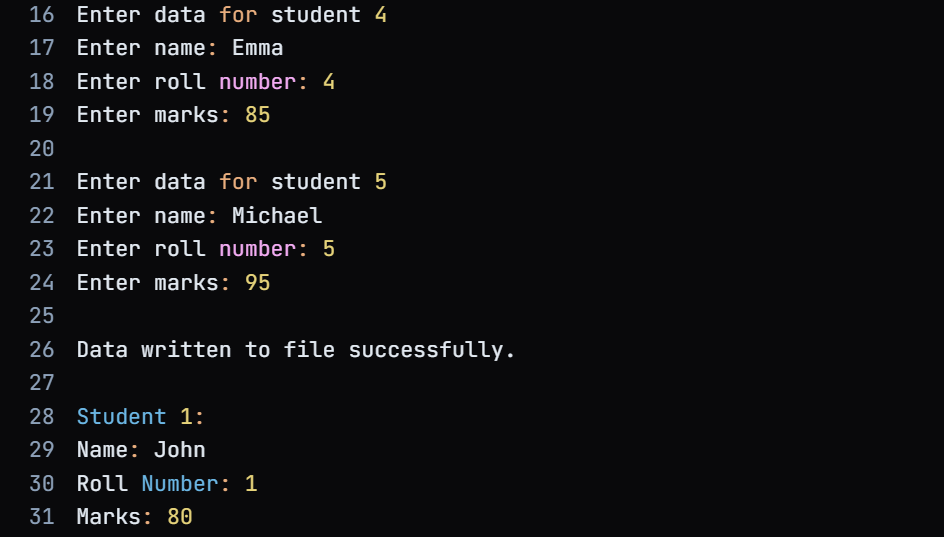
}

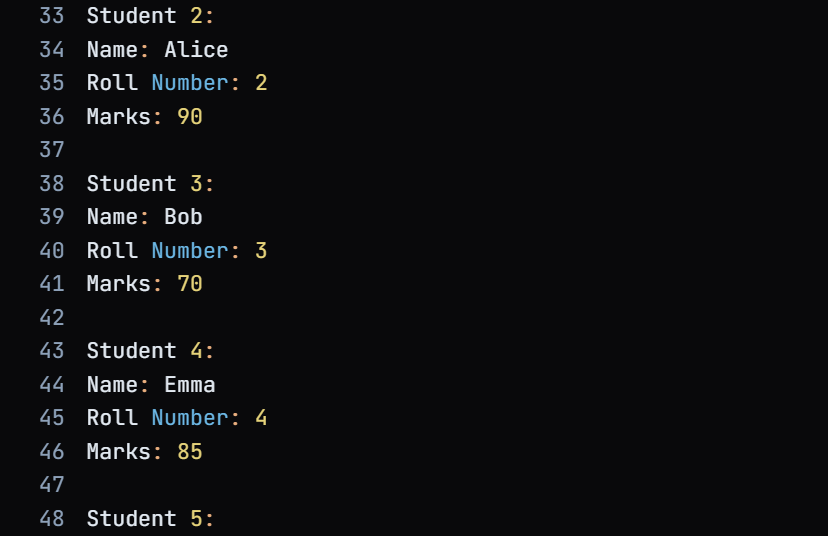
return 0;

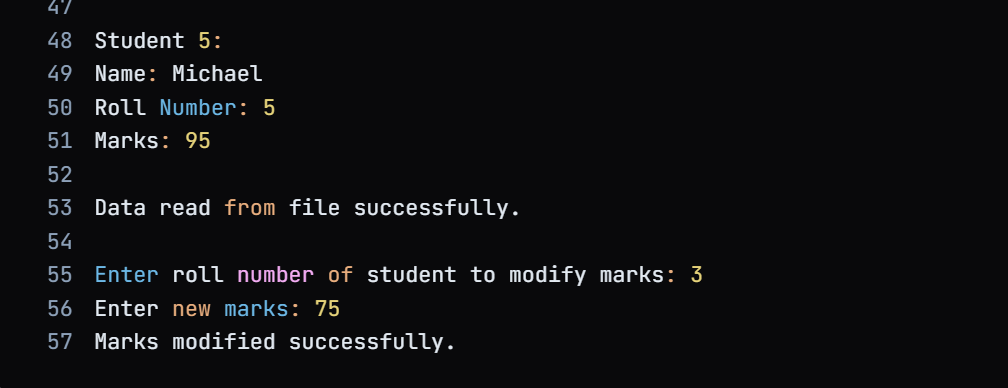
}

Output:









QUESTION # 2:

Part A:

**Create a base class Shape with data member color and member functions to set and get color. • Create a derived class Rectangle inheriting Shape with data members length and breadth. • Implement member functions to calculate area and perimeter of the rectangle. • Demonstrate constructor chaining in both base and derived classes.**

CODE:

#include <iostream>

#include <string>

class Shape {

protected:

string color;

public:

Shape(const string& c = "black") : color(c) {}

void setColor(const string& c) {

color = c;

}

string getColor() const {

return color;

}

};

class Rectangle : public Shape {

private:

int length;

int breadth;

public:

Rectangle(int l, int b, const string& c = "black")

: Shape(c), length(l), breadth(b) {}

int getLength() const {

return length;

}

int getBreadth() const {

return breadth;

}

int area() const {

return length \* breadth;

}

int perimeter() const {

return 2 \* (length + breadth);

}

};

int main() {

Rectangle rect(5, 3, "red");

cout << "Color: " << rect.getColor() << endl;

cout << "Length: " << rect.getLength() << endl;

cout << "Breadth: " << rect.getBreadth() << endl;

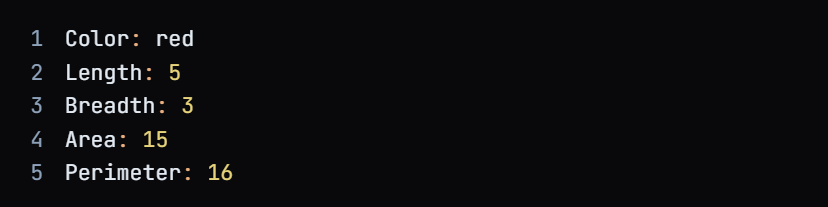
cout << "Area: " << rect.area() << endl;

cout << "Perimeter: " << rect.perimeter() << endl;

return 0;

}

Output:



Part B:

**Create a base class Animal with data member name. • Create a derived class Mammal inheriting Animal with data member numberOfLegs. • Create a derived class Dog inheriting Mammal with member functions to bark. • Demonstrate object creation and access to members of all classes**.

Code:

#include <iostream>

#include <string>

class Animal {

protected:

string name;

public:

Animal(const string& n) : name(n) {}

string getName() const {

return name;

}

};

class Mammal : public Animal {

protected:

int numberOfLegs;

public:

Mammal(const string& n, int legs) : Animal(n), numberOfLegs(legs) {}

int getNumberOfLegs() const {

return numberOfLegs;

}

};

class Dog : public Mammal {

public:

Dog(const string& n, int legs) : Mammal(n, legs) {}

void bark() const {

cout << "Woof! " << getName() << " is barking!" << endl;

}

};

int main() {

Dog myDog("Fido", 4);

cout << "Name: " << myDog.getName() << endl;

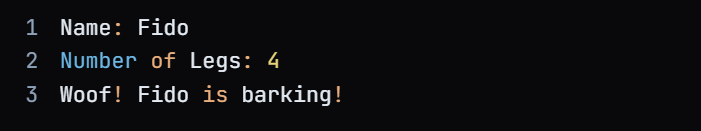
cout << "Number of Legs: " << myDog.getNumberOfLegs() << endl;

myDog.bark();

return 0;

}

Output:



Part C:

**Create a base class Person with data members name and address. • Create another base class Employee with data members employeeID and salary. • Create a derived class Teacher inheriting both Person and Employee with additional data member subject. • Demonstrate object creation and access to members of all classes. • Discuss the ambiguity problem in multiple inheritance and how to resolve it.**

Code:

#include <iostream>

#include <string>

class Person {

protected:

string name;

string address;

public:

Person(const string& n, const string& a) : name(n), address(a) {}

string getName() const {

return name;

}

string getAddress() const {

return address;

}

};

class Employee {

protected:

int employeeID;

double salary;

public:

Employee(int id, double s) : employeeID(id), salary(s) {}

int getEmployeeID() const {

return employeeID;

}

double getSalary() const {

return salary;

}

};

class Teacher : public Person, public Employee {

private:

string subject;

public:

Teacher(const string& n, const string& a, int id, double s, const string& sub)

: Person(n, a), Employee(id, s), subject(sub) {}

string getSubject() const {

return subject;

}

};

int main() {

Teacher myTeacher("John Doe", "123 Main St", 1234, 50000.0, "Math");

cout << "Name: " << myTeacher.getName() << endl;

cout << "Address: " << myTeacher.getAddress() << endl;

cout << "Employee ID: " << myTeacher.getEmployeeID() << endl;

cout << "Salary: " << myTeacher.getSalary() << endl;

cout << "Subject: " << myTeacher.getSubject() <<endl;

return 0;

}

Output:

