**Question no 1:**

**Task 1:**

#include <iostream>

#include <fstream>

using namespace std;

class Student {

public:

string name;

int roll\_number;

float marks;

// Function to input student data

void input\_data() {

cout << "Enter name: ";

cin >> name;

cout << "Enter roll number: ";

cin >> roll\_number;

cout << "Enter marks: ";

cin >> marks;

}

// Function to display student data

void display\_data() {

cout << "Name: " << name << ", Roll Number: " << roll\_number << ", Marks: " << marks << endl;

}

};

int main() {

const int num\_students = 5;

Student students[num\_students];

// Write data of 5 students to the file

ofstream outFile("students.txt");

for (int i = 0; i < num\_students; i++) {

cout << "\nEnter details for student " << i + 1 << ":\n";

students[i].input\_data();

outFile << students[i].name << " " << students[i].roll\_number << " " << students[i].marks << endl;

}

outFile.close();

// Read and display data from the file

ifstream inFile("students.txt");

cout << "\nDisplaying student data from the file:\n";

Student temp;

while (inFile >> temp.name >> temp.roll\_number >> temp.marks) {

temp.display\_data();

}

inFile.close();

// Modify the marks of a specific student

int roll\_to\_modify;

float new\_marks;

cout << "\nEnter roll number of the student to modify marks: ";

cin >> roll\_to\_modify;

cout << "Enter new marks: ";

cin >> new\_marks;

// Open the file for reading and writing to modify marks

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

while (file >> temp.name >> temp.roll\_number >> temp.marks) {

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

file.seekp(-sizeof(temp.marks), ios::cur); // Move back to modify marks

file << new\_marks;

break;

}

}

file.close();

// Display updated data

inFile.open("students.txt");

cout << "\nDisplaying updated student data from the file:\n";

while (inFile >> temp.name >> temp.roll\_number >> temp.marks) {

temp.display\_data();

}

inFile.close();

return 0;

}

**Task 2:**

**Part a:**

#include <iostream>

using namespace std;

// Base class Shape

class Shape {

public:

// Constructor

Shape(const string& color) : color(color) {}

// Set and get color

void setColor(const string& color) {

this->color = color;

}

string getColor() const {

return color;

}

private:

string color;

};

// Derived class Rectangle inheriting Shape

class Rectangle : public Shape {

public:

// Constructor with constructor chaining

Rectangle(const string& color, double length, double breadth)

: Shape(color), length(length), breadth(breadth) {}

// Calculate area and perimeter

double calculateArea() const {

return length \* breadth;

}

double calculatePerimeter() const {

return 2 \* (length + breadth);

}

private:

double length;

double breadth;

};

// Demonstration

int main() {

// Create a Rectangle object

Rectangle rect("blue", 4.0, 5.0);

// Accessing members and functions

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

cout << "Rectangle Area: " << rect.calculateArea() << endl;

cout << "Rectangle Perimeter: " << rect.calculatePerimeter() << endl;

return 0;

}

**Task 2:**

**Part b:**

#include <iostream>

#include <string>

using namespace std;

// Base class Animal

class Animal {

public:

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

string getName() const {

return name;

}

protected:

string name;

};

// Derived class Mammal inheriting Animal

class Mammal : public Animal {

public:

Mammal(const string& name, int numberOfLegs)

: Animal(name), numberOfLegs(numberOfLegs) {}

int getNumberOfLegs() const {

return numberOfLegs;

}

protected:

int numberOfLegs;

};

// Derived class Dog inheriting Mammal

class Dog : public Mammal {

public:

Dog(const string& name, int numberOfLegs)

: Mammal(name, numberOfLegs) {}

string bark() const {

return name + " says Woof!";

}

};

// Demonstration

int main() {

Dog dog("Buddy", 4);

// Accessing members from all classes

cout << "Dog's Name: " << dog.getName() << endl;

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

cout << dog.bark() << endl;

return 0;

}

**Task 2:**

**Part c:**

#include <iostream>

#include <string>

using namespace std;

// Base class Person

class Person {

public:

Person(const string& name, const string& address)

: name(name), address(address) {}

void setName(const string& name) {

this->name = name;

}

string getName() const {

return name;

}

void setAddress(const string& address) {

this->address = address;

}

string getAddress() const {

return address;

}

private:

string name;

string address;

};

// Base class Employee

class Employee {

public:

Employee(int employeeID, double salary)

: employeeID(employeeID), salary(salary) {}

void setEmployeeID(int employeeID) {

this->employeeID = employeeID;

}

int getEmployeeID() const {

return employeeID;

}

void setSalary(double salary) {

this->salary = salary;

}

double getSalary() const {

return salary;

}

private:

int employeeID;

double salary;

};

// Derived class Teacher inheriting Person and Employee

class Teacher : public Person, public Employee {

public:

Teacher(const string& name, const string& address, int employeeID, double salary, const string& subject)

: Person(name, address), Employee(employeeID, salary), subject(subject) {}

void setSubject(const string& subject) {

this->subject = subject;

}

string getSubject() const {

return subject;

}

private:

string subject;

};

// Demonstration

int main() {

Teacher teacher("Jane Smith", "456 Elm St", 1234, 60000.0, "Physics");

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

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

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

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

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

return 0;

}