CSS-452 Assignment-9

INDRAJIT DAS

CS-X 22CS8024

Q1. Write a program to declare a matrix class which has a data member integer array as 3X3.

Derive class matrixA from class matrix and matrixB from matrixA. All these classes should have a function show() to display the contents. Read and display the elements of all three matrices.

Cpp Code:

```cpp
#include <iostream>
```

```
using namespace std;



//base class named Matrix

class Matrix {

protected:

   int data[3][3]; // Protected member variable to store 3x3 matrix data

public:

   // Public member function to display the matrix

   void show() {

       cout << "Matrix:" << endl;

       // Loop through each row

       for (int i = 0; i < 3; ++i) {

           // Loop through each column

           for (int j = 0; j < 3; ++j) {

               cout << data[i][j] << " "; // Display each element of the
matrix

           }
```

```
            cout << endl; // Move to the next line after displaying each
row

        }

    }

};



// Derived class MatrixA from Matrix

class MatrixA : public Matrix {

public:

    // Function to read elements for MatrixA

    void read() {

        cout << "Enter elements for MatrixA:" << endl;

        // Loop through each row

        for (int i = 0; i < 3; ++i) {

            // Loop through each column

            for (int j = 0; j < 3; ++j) {

                cin >> data[i][j]; // Read each element entered by the user
into the data array
```

```
            }

        }

    }

};



// Derived class MatrixB from MatrixA

class MatrixB : public MatrixA {

public:

    // Function to read elements for MatrixB

    void read() {

        cout << "Enter elements for MatrixB:" << endl;

        // Loop through each row

        for (int i = 0; i < 3; ++i) {

            // Loop through each column

            for (int j = 0; j < 3; ++j) {

                cin >> data[i][j]; // Read each element entered by the user
into the data array
```

```
            }

        }

    }

};



// Main function

int main() {

    Matrix mat; // Create an instance of Matrix

    MatrixA matA; // Create an instance of MatrixA

    MatrixB matB; // Create an instance of MatrixB



    cout << "Displaying Matrix: (no initialisation)  <---- Base Class" <<
endl;

    mat.show(); // Display the initial empty matrix



    cout << "\nReading MatrixA:" << endl;

    matA.read(); // Read elements for MatrixA
```

```
    cout << "\nDisplaying MatrixA:" << endl;

    matA.show(); // Display MatrixA



    cout << "\nReading MatrixB:" << endl;

    matB.read(); // Read elements for MatrixB

    cout << "\nDisplaying MatrixA:" << endl;

    matB.show(); // Display MatrixB



    return 0;

}
```

Output:

```
❯ g++ 1.cpp
❯ ./a.out
Displaying Matrix: (no initialisation)  <---- Base Class
Matrix:
135168 0 -1704481792
1951463202 159813216 30330
155184288 30330 4607

Reading MatrixA:
Enter elements for MatrixA:
1 2 3
4 5 6
7 8 9

Displaying MatrixA:
Matrix:
1 2 3
4 5 6
7 8 9

Reading MatrixB:
Enter elements for MatrixB:
3 4 5
7 8 9
1 2 3

Displaying MatrixA:
Matrix:
3 4 5
7 8 9
1 2 3
```

Q2) Create a generic base class called vehicle that stores number of wheels and speed. Create

the following derived class:

a. Car that inherits vehicle and also stores number of passengers

b. Truck that inherits vehicle and also stores load limit

c. Write a program to create objects of these classes and display all the information

about the car and truck. Also compare the speed of the vehicles and display 'faster'

or 'slower' if the car is slower than the truck.

Cpp Code

```cpp
#include <iostream>
#include <string>

using namespace std;

// Base class Vehicle
class Vehicle {
protected:
   int wheels;
   double speed;
```

```
public:
    // Constructor to initialize the wheels and speed
    Vehicle(int _wheels, double _speed) : wheels(_wheels), speed(_speed) {}

    // Function to display vehicle information
    void displayInfo() {
        cout << "Number of wheels: " << wheels << endl;
        cout << "Speed: " << speed << " km/h" << endl;
    }

    // Function to compare speed with another vehicle
    void compareSpeed(Vehicle& other) {
        if (speed > other.speed)
            cout << "Faster";
        else if (speed < other.speed)
            cout << "Slower";
        else
            cout << "Equal";
    }
};

// Derived class Car inheriting from Vehicle
class Car : public Vehicle {
private:
    int passengers;
```

```
public:
    // Constructor to initialize the Car with number of wheels, speed, and passengers
    Car(int _wheels, double _speed, int _passengers) : Vehicle(_wheels, _speed), passengers(_passengers) {}

    // Function to display car information
    void displayCarInfo() {
        displayInfo(); // Call base class function to display vehicle information
        cout << "Number of passengers: " << passengers << endl;
    }
};

// Derived class Truck inheriting from Vehicle
class Truck : public Vehicle {
private:
    double loadLimit;
public:
    // Constructor to initialize the Truck with number of wheels, speed, and load limit
    Truck(int _wheels, double _speed, double _loadLimit) : Vehicle(_wheels, _speed), loadLimit(_loadLimit) {}

    // Function to display truck information
    void displayTruckInfo() {
        displayInfo(); // Call base class function to display vehicle
```

```
information
        cout << "Load limit: " << loadLimit << " tons" << endl;
    }
};


int main() {
    // Create objects of Car and Truck classes
    Car car(4, 180.0, 4); // Car with 4 wheels, speed 180 km/h, and 4
passengers
    Truck truck(6, 40.0, 20.0); // Truck with 6 wheels, speed480 km/h, and
load limit 20 tons
    // Display information about the car
    cout << "Car Information:" << endl;
    car.displayCarInfo();
    cout << endl;


    // Display information about the truck
    cout << "Truck Information:" << endl;
    truck.displayTruckInfo();
    cout << endl;


    // Compare the speed of the car and the truck
    cout << "Comparing speed:" << endl;
    cout << "Car's speed is ";
    car.compareSpeed(truck);
    cout << " than truck" << endl;
```

```
    return 0;
}
```

Output:

```
❯ g++ 2.cpp
❯ ./a.out
Car Information:
Number of wheels: 4
Speed: 180 km/h
Number of passengers: 4

Truck Information:
Number of wheels: 6
Speed: 40 km/h
Load limit: 20 tons

Comparing speed:
Car's speed is Faster than truck
```

Q3) Rock Paper Scissor Implementation using virtual Class

Cpp Code:

```cpp
#include <iostream>

using namespace std;


class Tool {

protected:

   int strength;

   char type;

public:

   Tool(int s, char t) : strength(s), type(t) {}

   int getStrength() { return strength; }  // Getter function for strength

   char getType() { return type; }  // Getter function for type

   virtual bool fight(Tool* tool) = 0;  // Pure virtual function

};


class Rock : public Tool {

public:

   Rock(int s) : Tool(s, 'r') {}

   bool fight(Tool* tool) {

       int newStrength = strength;

       if (tool->getType() == 's') newStrength *= 2;

       else if (tool->getType() == 'p') newStrength /= 2;
```

```
        return newStrength > tool->getStrength();
    }
};


class Paper : public Tool {
public:
    Paper(int s) : Tool(s, 'p') {}
    bool fight(Tool* tool) {
        int newStrength = strength;
        if (tool->getType() == 'r') newStrength *= 2;
        else if (tool->getType() == 's') newStrength /= 2;
        return newStrength > tool->getStrength();
    }
};


class Scissors : public Tool {
public:
    Scissors(int s) : Tool(s, 's') {}
    bool fight(Tool* tool) {
        int newStrength = strength;
        if (tool->getType() == 'p') newStrength *= 2;
        else if (tool->getType() == 'r') newStrength /= 2;
        return newStrength > tool->getStrength();
    }
};
```

```cpp
int main() {
   Scissors s1(5);
   Paper p1(7);
   Rock r1(15);
   cout << s1.fight(&p1) <<" "<< p1.fight(&s1) << endl;
   cout << p1.fight(&r1) <<" "<< r1.fight(&p1) << endl;
   cout << r1.fight(&s1) <<" "<< s1.fight(&r1) << endl;
   return 0;
}
```

Output:

```
❯ g++ 3.cpp
❯ ./a.out
1 0
0 0
1 0
```