ASSIGNMENT-10

Name-Md Asif

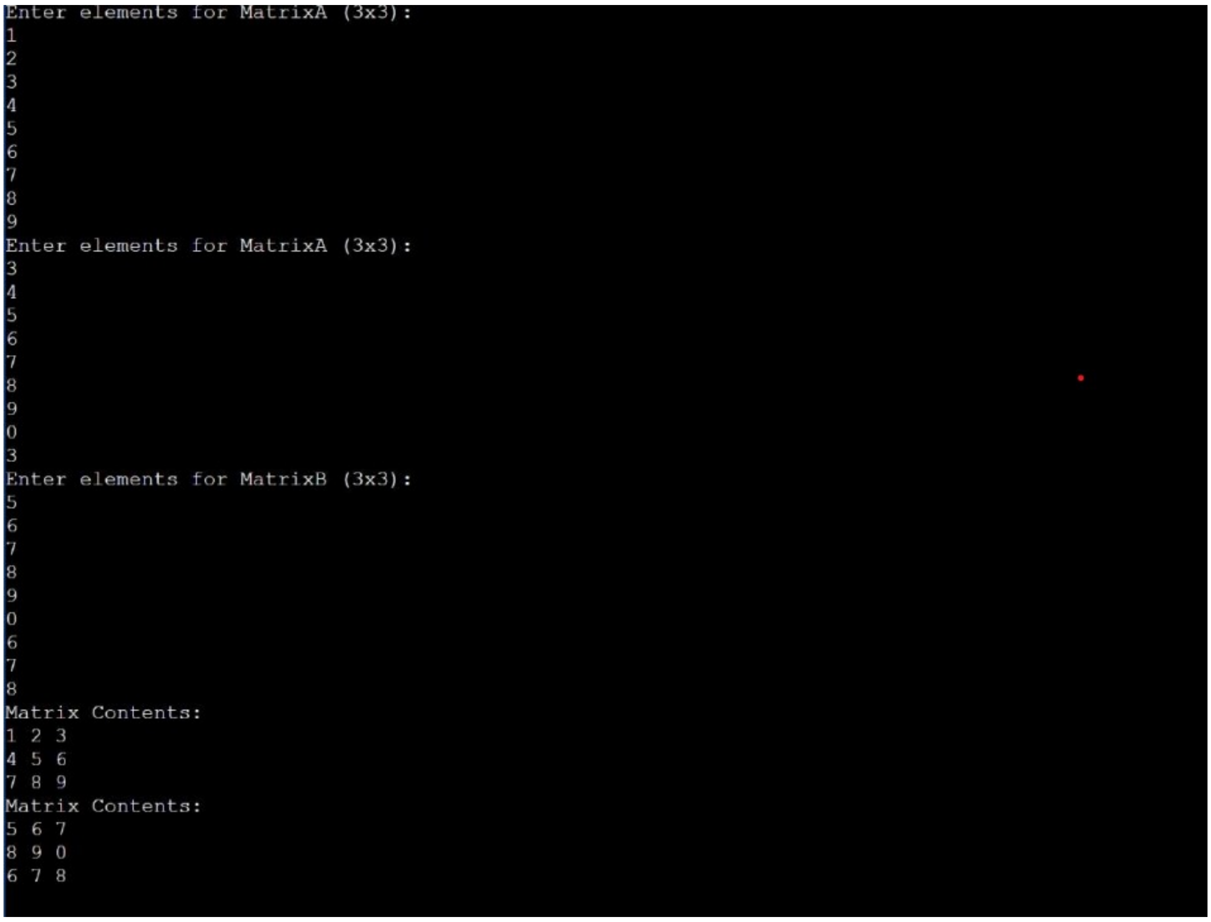
Roll no- 21CS8022

==

1)

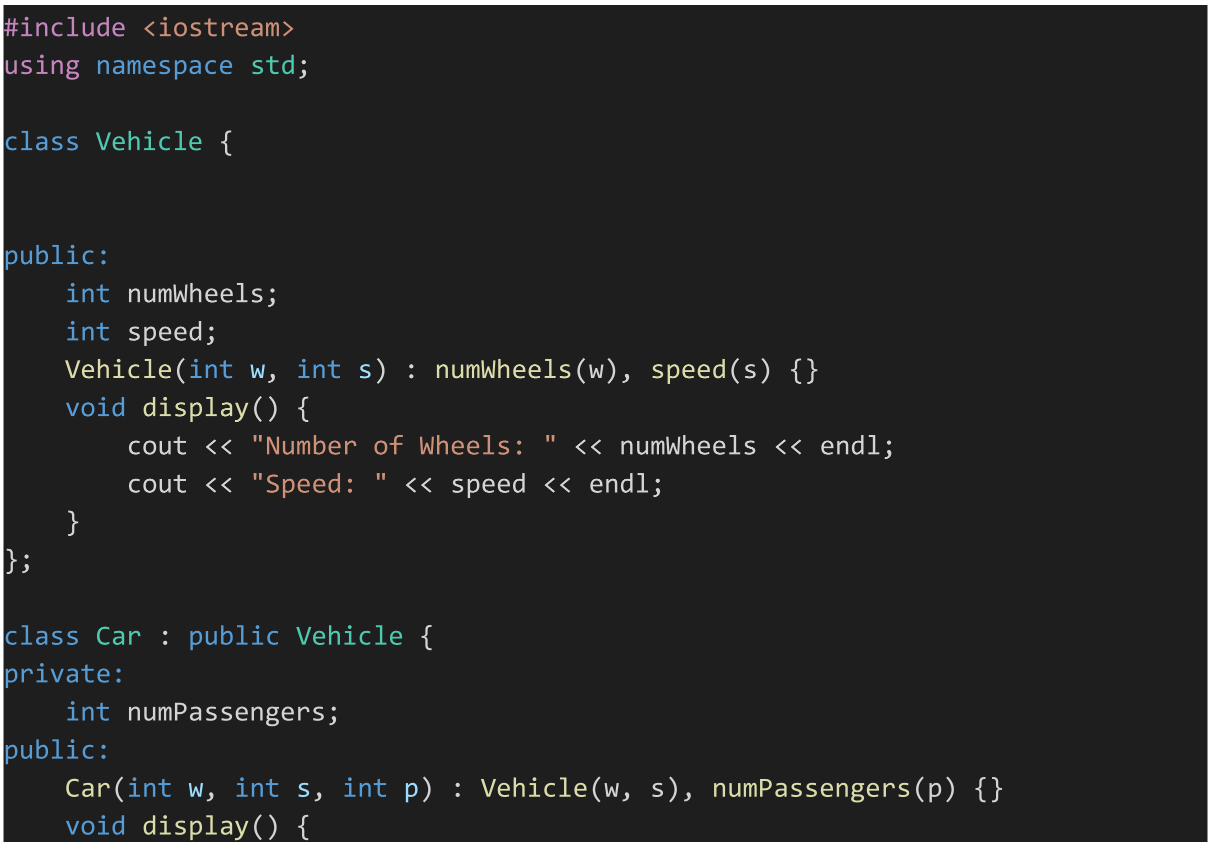
```cpp
#include<iostream>
using namespace std;

class matrix {
    protected:
        int arr[3][3];
    public:
        void read() {
            cout<<"Enter the elements of the matrix:"<<endl;
            for(int i=0; i<3; i++) {
                for(int j=0; j<3; j++) {
                    cin>>arr[i][j];
                }
            }
        }
        void show() {
            cout<<"Matrix:"<<endl;
            for(int i=0; i<3; i++) {
                for(int j=0; j<3; j++) {
                    cout<<arr[i][j]<<" ";
                }
                cout<<endl;
            }
        }
};

class matrixA : public matrix {
    public:
        void add(int n) {
            for(int i=0; i<3; i++) {
                for(int j=0; j<3; j++) {
                    arr[i][j] += n;
                }
            }
        }
};

class matrixB : public matrixA {
    public:
        void sub(int n) {
            for(int i=0; i<3; i++) {
                for(int j=0; j<3; j++) {
                    arr[i][j] -= n;
                }
            }
        }
};

int main() {
    matrix m;
    matrixA mA;
    matrixB mB;

    m.read();
    m.show();

    mA.read();
    mA.show();
    mA.add(5);
    mA.show();

    mB.read();
    mB.show();
    mB.add(10);
    mB.show();
    mB.sub(3);
    mB.show();

return 0;
}
```

OUTPUT:



```
Enter elements for MatrixA (3x3):
1
2
3
4
5
6
7
8
9
Enter elements for MatrixA (3x3):
3
4
5
6
7
8
9
0
3
Enter elements for MatrixB (3x3):
5
6
7
8
9
0
6
7
8
Matrix Contents:
1 2 3
4 5 6
7 8 9
Matrix Contents:
5 6 7
8 9 0
6 7 8
```

2)



```
#include <iostream>
using namespace std;

class Vehicle {


public:
    int numWheels;
    int speed;
    Vehicle(int w, int s) : numWheels(w), speed(s) {}
    void display() {
        cout << "Number of Wheels: " << numWheels << endl;
        cout << "Speed: " << speed << endl;
    }
};

class Car : public Vehicle {
private:
    int numPassengers;
public:
    Car(int w, int s, int p) : Vehicle(w, s), numPassengers(p) {}
    void display() {
```

```cpp
        Vehicle::display();
        cout << "Number of Passengers: " << numPassengers << endl;
    }
};

class Truck : public Vehicle {
private:
    int loadLimit;
public:
    Truck(int w, int s, int l) : Vehicle(w, s), loadLimit(l) {}
    void display() {
        Vehicle::display();
        cout << "Load Limit: " << loadLimit << endl;
    }
};

int main() {
    Car c(4, 120, 5);
    Truck t(6, 80, 5000);

    cout << "Car Details:" << endl;
    c.display();

    cout << "\nTruck Details:" << endl;
    t.display();

    if (c.speed < t.speed) {
        cout << "\nTruck is faster than Car." << endl;
    }
    else {
        cout << "\nCar is faster than Truck." << endl;
    }

return 0;
}
```

OUTPUT:

```
Car Details:
Number of Wheels: 4
Speed: 120
Number of Passengers: 5

Truck Details:
Number of Wheels: 6
Speed: 80
Load Limit: 5000
```

```
Car is faster than Truck.
```

3)

```
#include <iostream>
using namespace std;

class Tool
{
protected:
    int strength; // strength of the tool
    char type; // type of the tool ('r' for Rock, 'p' for Paper, 's' for
Scissors)

public:
    Tool(int strength, char type) : strength(strength), type(type) {}

    void setStrength(int strength) {
        this->strength = strength;
    }

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

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

    // Function to compare strengths of tools
    virtual bool fight(Tool opponent) {
        return strength > opponent.strength;
    }
};

class Rock : public Tool
{
public:
    Rock(int strength) : Tool(strength, 'r') {}

    // Override fight function to implement Rock's strength advantage
```

```cpp
    bool fight(Tool opponent) override {
        if (opponent.getType() == 's') { // Rock vs Scissors
            return strength * 2 > opponent.getStrength();
        } else if (opponent.getType() == 'p') { // Rock vs Paper
            return strength / 2 > opponent.getStrength();
        } else {
            return strength > opponent.getStrength();
        }
    }
};

class Paper : public Tool
{
public:
    Paper(int strength) : Tool(strength, 'p') {}

    // Override fight function to implement Paper's strength advantage
    bool fight(Tool opponent) override {
        if (opponent.getType() == 'r') { // Paper vs Rock
            return strength * 2 > opponent.getStrength();
        } else if (opponent.getType() == 's') { // Paper vs Scissors
            return strength / 2 > opponent.getStrength();
        } else {
            return strength > opponent.getStrength();
        }
    }
};

class Scissors : public Tool
{
public:
    Scissors(int strength) : Tool(strength, 's') {}

    // Override fight function to implement Scissors' strength advantage
    bool fight(Tool opponent) override {
        if (opponent.getType() == 'p') { // Scissors vs Paper
            return strength * 2 > opponent.getStrength();
        } else if (opponent.getType() == 'r') { // Scissors vs Rock
            return strength / 2 > opponent.getStrength();
        } else {
            return strength > opponent.getStrength();
        }
    }
};

int main() {
    // Example main function
    // You may add your own testing code if you like
    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:

```
10
00
10
```