

Name: Diptangshu Dey

Roll No. 20CS8018

Assignment-6

1. Code:

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

class Matrix {
protected:
    int val[3][3];

public:
    void show();
    void read();
};

void Matrix::read() {
    cout<<"Enter values for the matrix: "<<endl;
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout << "[" << i << "]"[" << j << "]: ";
            cin>>val[i][j];
        }
    }
}

void Matrix::show() {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            cout<<val[i][j]<<" ";
        }
        cout<<endl;
    }
}

class MatrixA : public Matrix {
public:
    void show();
};

void MatrixA::show() {
    Matrix::show();
}

class MatrixB : public MatrixA {
public:
    void show();
};
```

```
void MatrixB::show() {
    MatrixA::show();
}

int main() {
    Matrix m1;
    MatrixA m2;
    MatrixB m3;

    cout << "For m1: " << endl;
    m1.read();

    cout << "For m2: " << endl;
    m2.read();

    cout << "For m3: " << endl;
    m3.read();

    cout << endl << "m1: " << endl;
    m1.show();

    cout << endl << "m2: " << endl;
    m2.show();

    cout << endl << "m3: " << endl;
    m3.show();

    return 0;
}
```

Output:

```
Matrix.cpp:28:   cout<<endl;
Matrix.cpp:29: }
Matrix.cpp:30: }
```

```
diptangshudey@ArchACER: ~/.../CS5452_DSA_assign/Assign6 [master] ./Matrix
For m1:
Enter values for the matrix:
[0][0]: 1
[0][1]: 2
[0][2]: 3
[1][0]: 4
[1][1]: 5
[1][2]: 6
[2][0]: 7
[2][1]: 8
[2][2]: 9
For m2:
Enter values for the matrix:
[0][0]: 11
[0][1]: 12
[0][2]: 13
[1][0]: 14
[1][1]: 15
[1][2]: 16
[2][0]: 17
[2][1]: 18
[2][2]: 19
For m3:
Enter values for the matrix:
[0][0]: 21
[0][1]: 22
[0][2]: 23
[1][0]: 24
[1][1]: 25
[1][2]: 26
[2][0]: 27
[2][1]: 28
[2][2]: 29

m1:
1 2 3
4 5 6
7 8 9

m2:
11 12 13
14 15 16
17 18 19

m3:
21 22 23
24 25 26
27 28 29
diptangshudey@ArchACER: ~/.../CS5452_DSA_assign/Assign6 [master] 
```

2. Code:

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

```
template <class T> class vehicle {
protected:
    T wheel;
```

```
public:
    T speed;
    vehicle(T w = 0, T s = 0) {
        wheel = w;
        speed = s;
    }
    void input();
    void show();
};
```

```
template <class T> class truck : public vehicle<T> {
protected:
    T load;
```

```
public:
    void input();
    void show();
};
```

```
template <class T> class car : public vehicle<T> {
protected:
    T pass;
```

```

public:
    void input();
    void show();
    void isFast(truck<T> t);
};

template <class T> void vehicle<T>::input() {
    cout << "Enter number of wheels: ";
    cin >> wheel;
    cout << "Enter Speed: ";
    cin >> speed;
}

template <class T> void vehicle<T>::show() {
    cout << "Number of wheels in vehicle: " << wheel << endl;
    cout<<"Speed of vehicle: "<<speed<<endl;
}

template <class T> void car<T>::input() {
    vehicle<T>::input();
    cout << "Enter number of passengers: ";
    cin >> pass;
}

template <class T> void car<T>::show() {
    vehicle<T>::show();
    cout<<"Number of passengers in Car is: "<<pass<<endl;
}

template <class T> void car<T>::isFast(truck<T> t) {
    if (vehicle<T>::speed > t.speed) {
        cout<<"faster"<<endl;
    } else if (vehicle<T>::speed < t.speed) {
        cout<<"slower"<<endl;
    } else {
        cout << "same " << endl;
    }
}

template <class T> void truck<T>::input() {
    vehicle<T>::input();
    cout << "Enter maximum load of truck: ";
    cin >> load;
}

template <class T> void truck<T>::show() {
    vehicle<T>::show();
    cout<<"Maximum load of truck is: "<<load<<endl;
}

```

```

int main() {
    car<int> c;
    truck<int> t;

    cout<<"For Car: "<<endl;
    c.input();

    cout << "For Truck: " << endl;
    t.input();

    c.show();
    t.show();
    c.isFast(t);
    return 0;
}

```

Output:

The screenshot shows a Visual Studio Code editor with a C++ file named `Vehicle.cpp` open. The code defines a `vehicle` template class with a `protected` `wheel` member and a `public` `speed` member. It also includes a `show` method that prints the vehicle's details. The terminal output shows the program being run twice. In the first run, a car is created with 4 wheels and a speed of 50, and a truck is created with 8 wheels and a speed of 10. In the second run, a car is created with 4 wheels and a speed of 40, and a truck is created with 8 wheels and a speed of 900. The output also shows the number of passengers in each vehicle.

```

diptangshudey@ArchACER: ~/.../CS5452_DSA_assign/Assign6 [master] ./veh
For Car:
Enter number of wheels: 4
Enter Speed: 50
Enter number of passengers: 2
For Truck:
Enter number of wheels: 8
Enter Speed: 10
Enter maximum load of truck: 20
Number of wheels in vehicle: 4
Speed of vehicle: 50
Number of passengers in Car is: 2
Number of wheels in vehicle: 8
Speed of vehicle: 10
Maximum load of truck is: 20
Faster
diptangshudey@ArchACER: ~/.../CS5452_DSA_assign/Assign6 [master] ./veh
For Car:
Enter number of wheels: 4
Enter Speed: 40
Enter number of passengers: 4
For Truck:
Enter number of wheels: 8
Enter Speed: 900
Enter maximum load of truck: 900
Number of wheels in vehicle: 4
Speed of vehicle: 40
Number of passengers in Car is: 4
Number of wheels in vehicle: 8
Speed of vehicle: 60
Maximum load of truck is: 900
Slower
diptangshudey@ArchACER: ~/.../CS5452_DSA_assign/Assign6 [master]

```

3. Code:

```

#include <iostream>
using namespace std;

class Tool
{
protected:
    int strength;
    char type;

```

```

public:
    void setStrength(int s) { strength = s; }
    bool fight(Tool t);
};

bool Tool::fight(Tool t) {
    if ((type == 'r' && t.type == 'p') || (type == 'p' && t.type == 's') || (type == 's' && t.type == 'r'))
    {
        if (strength / 2 > t.strength) {
            return true;
        } else {
            return false;
        }
    } else if ((type == 'r' && t.type == 's') || (type == 's' && t.type == 'p') ||
                (type == 'p' && t.type == 'r')) {
        if (strength * 2 > t.strength) {
            return true;
        } else {
            return false;
        }
    }
}

if (strength > t.strength) {
    return true;
} else {
    return false;
}
};

/*Implement class Scissors */
class Scissors : public Tool {
public:
    Scissors(int s) {
        Tool::setStrength(s);
        type = 's';
    }
};

/*Implement class Paper */
class Paper : public Tool {
public:
    Paper(int s) {
        Tool::setStrength(s);
        type = 'p';
    }
};

/*Implement class Rock */
class Rock : public Tool {
public:

```

```

Rock(int s) {
    Tool::setStrength(s);
    type = 'r';
}
};

```

```

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:

```

rps.cpp
1  Tool::setStrength(s);
2  type = 'p';
3  };
4
5  /**Implement class Rock */
6  class Rock : public Tool {
7  public:
8      Rock(int s) {
9          Tool::setStrength(s);
10         type = 'r';
11     }
12 };
13
14
15 int main() {
16
17     // Example main function
18
19     // You may add your own testing code if you like
20
21     Scissors s1(5);
22
23     Paper p1(7);
24
25     Rock r1(15);
26
27     cout << s1.fight(p1) << p1.fight(s1) << endl;
28
29     cout << p1.fight(r1) << r1.fight(p1) << endl;
30
31     cout << r1.fight(s1) << s1.fight(r1) << endl;
32
33     return 0;
34 }

```

```

diptangshudey@ArchACER: ~/.../CS5452_DSA_assign/Assign6 [master] ./rps
10
00
10
diptangshudey@ArchACER: ~/.../CS5452_DSA_assign/Assign6 [master]

```