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();

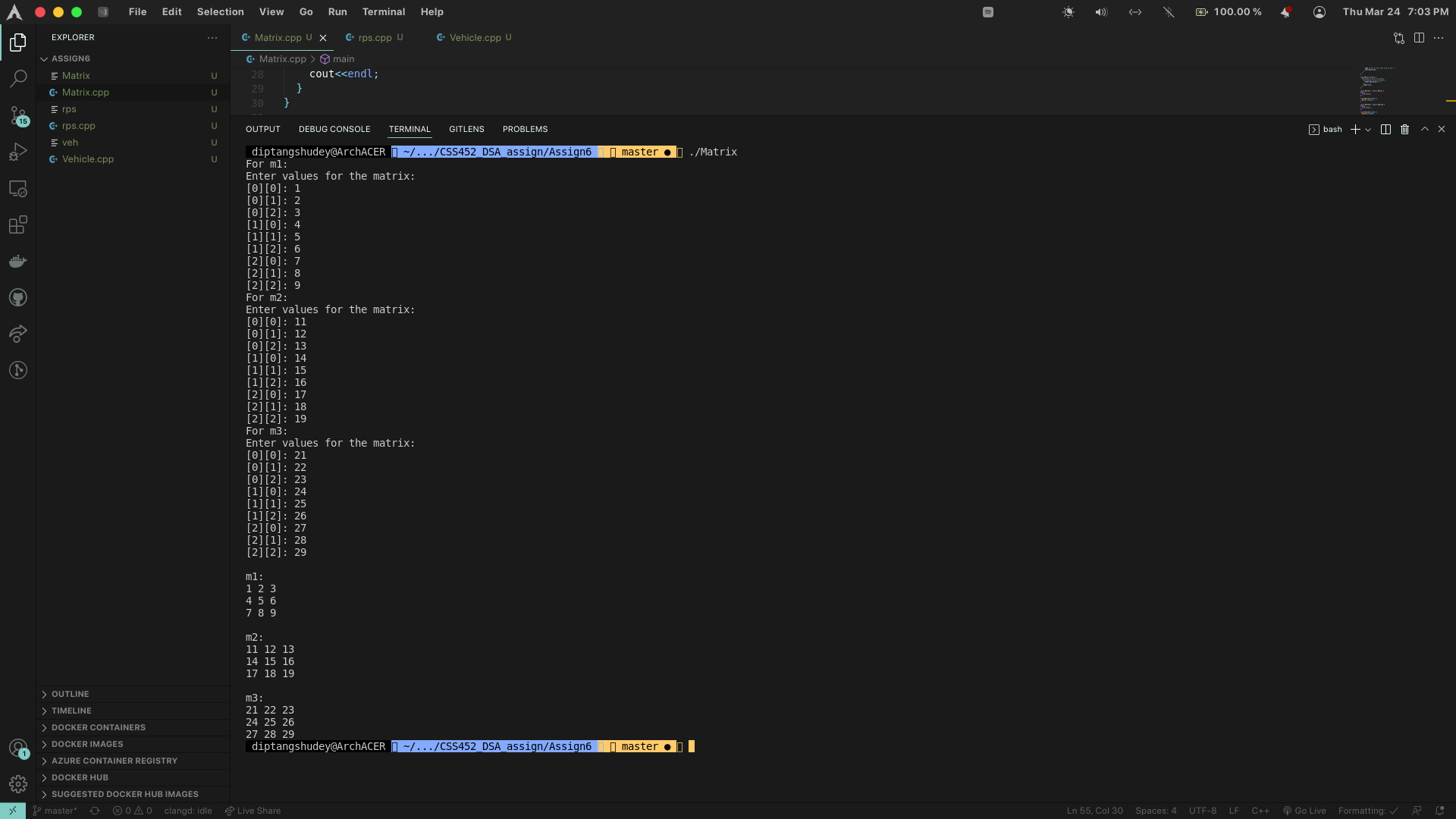
cout << endl << "m3: " << endl;

m3.show();

*return* 0;

}

Output:



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();

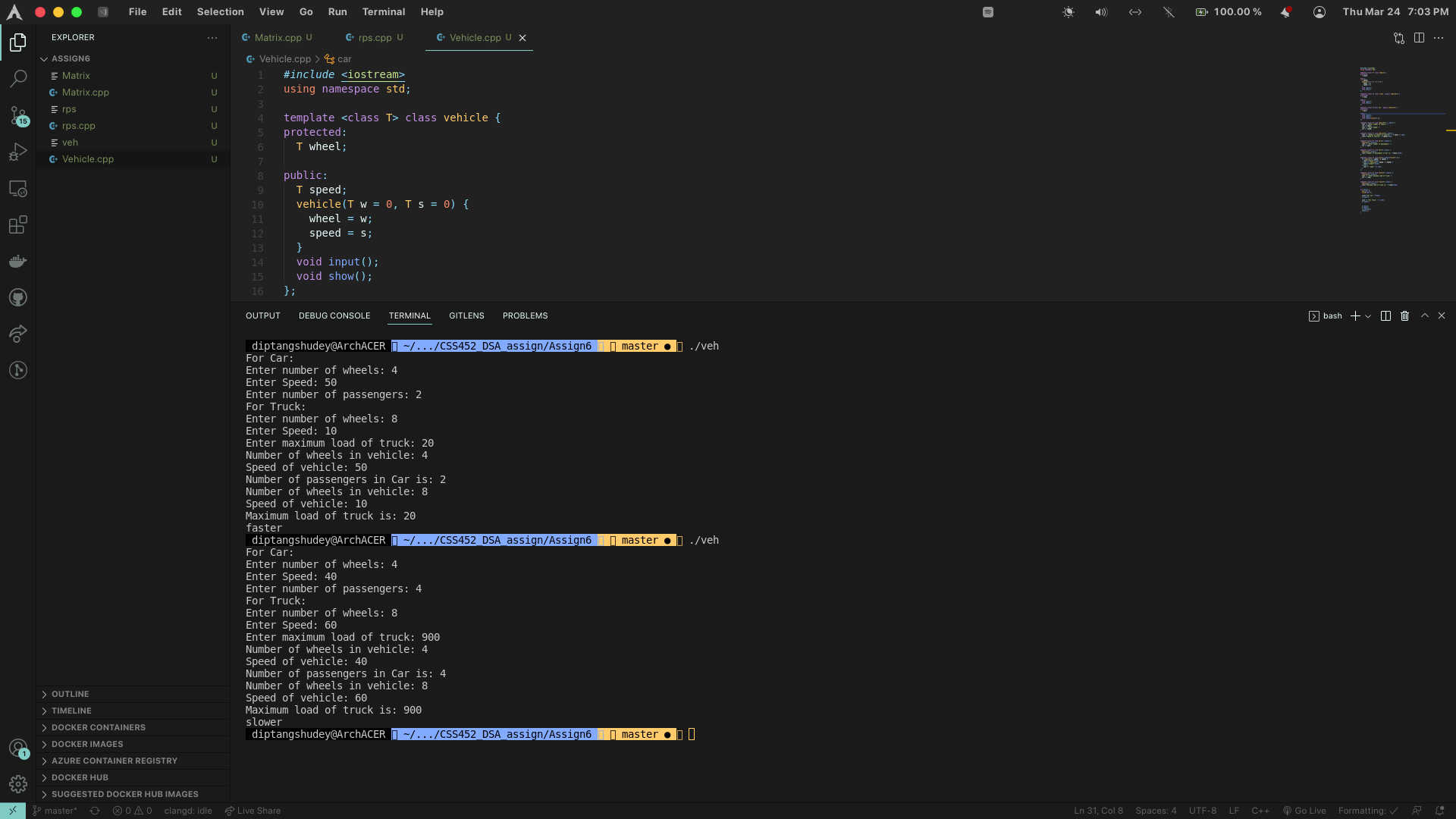
t.show();

c.isFast(t);

*return* 0;

}

Output:



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:

