**TUT-5**

**NAME-SANCHIT JAIN**

**BATCH – B-7**

**ENROLL – 21103192**

**1.**

(i) No Output As every static member must be initialized and we have initialized variable ‘a’ so no run time error. Also as variable ‘a’ is a static member and is referenced using the class for initialization therefore no compiler error.

(ii) Count:6

Count:7

(iii) OUTPUT:

new called

Constructor called

Destructor called

delete called

There are two things that happen in the above statement--memory allocation and object construction; the new keyword is responsible for both. One step in the process is to call operator new in order to allocate memory; the other step is to actually invoke the constructor. Operator new only allows us to change the memory allocation method, but does not do anything with the constructor calling method.Keyword new is responsible for calling the constructor, not operator new.

(iv) OUTPUT: 10

20

Class B has as conversion operator overloaded, so an object of B can be converted to that of A. Also, class A has a constructor which can be called with single integer argument, so an int can be converted to A

(v) OUTPUT: x=3,y=2;

This is a simple example of function call operator overloading

2. #include <iostream>

using namespace std;

class time

{

    int hr;

    int min;

    int sec;

public:

    time()

    {

        hr = min = sec = 0;

    }

    time(int h, int m, int s)

    {

        hr = h;

        min = m;

        sec = s;

    }

    friend bool operator==(time &t1, time &t2);

};

bool operator==(time &t1, time &t2)

{

    if (t1.hr == t2.hr && t1.min == t2.min && t1.sec == t2.sec)

    {

        return true;

    }

    else

    {

        return false;

    }

}

int main()

{

    time t1(10, 20, 30);

    time t2(9, 50, 40);

    if (t1 == t2)

    {

        cout << "Time are equal" << endl;

    }

    else

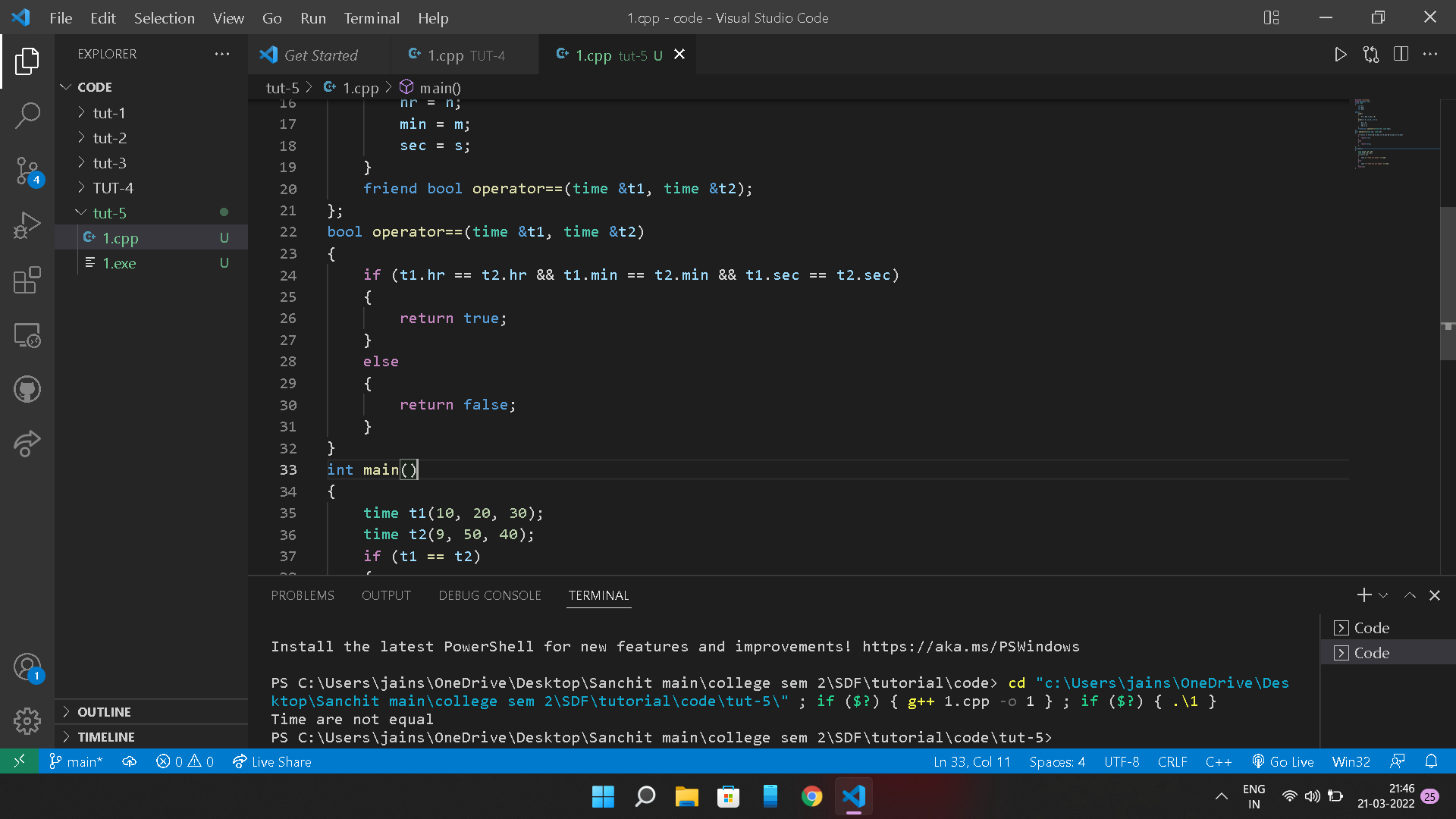
    {

        cout << "Time are not equal" << endl;

    }

    return 0;

}



**2.**

#include <iostream>

using namespace std;

class Distance

{

    int feet;

    int inches;

public:

    Distance()

    {

        feet = 0;

        inches = 0;

    }

    Distance(int f, int i)

    {

        feet = f;

        inches = i;

    }

    friend ostream &operator<<(ostream &output, const Distance &D)

    {

        output << "Feet=" << D.feet << "Inches=" << D.inches << endl;

        return output;

    }

    friend istream &operator>>(istream &input, Distance &D)

    {

        input >> D.feet >> D.inches;

        return input;

    }

};

int main()

{

    Distance D1(10, 20), D2(11, 36), D3;

    cout << "Enter the third distance" << endl;

    cin >> D3;

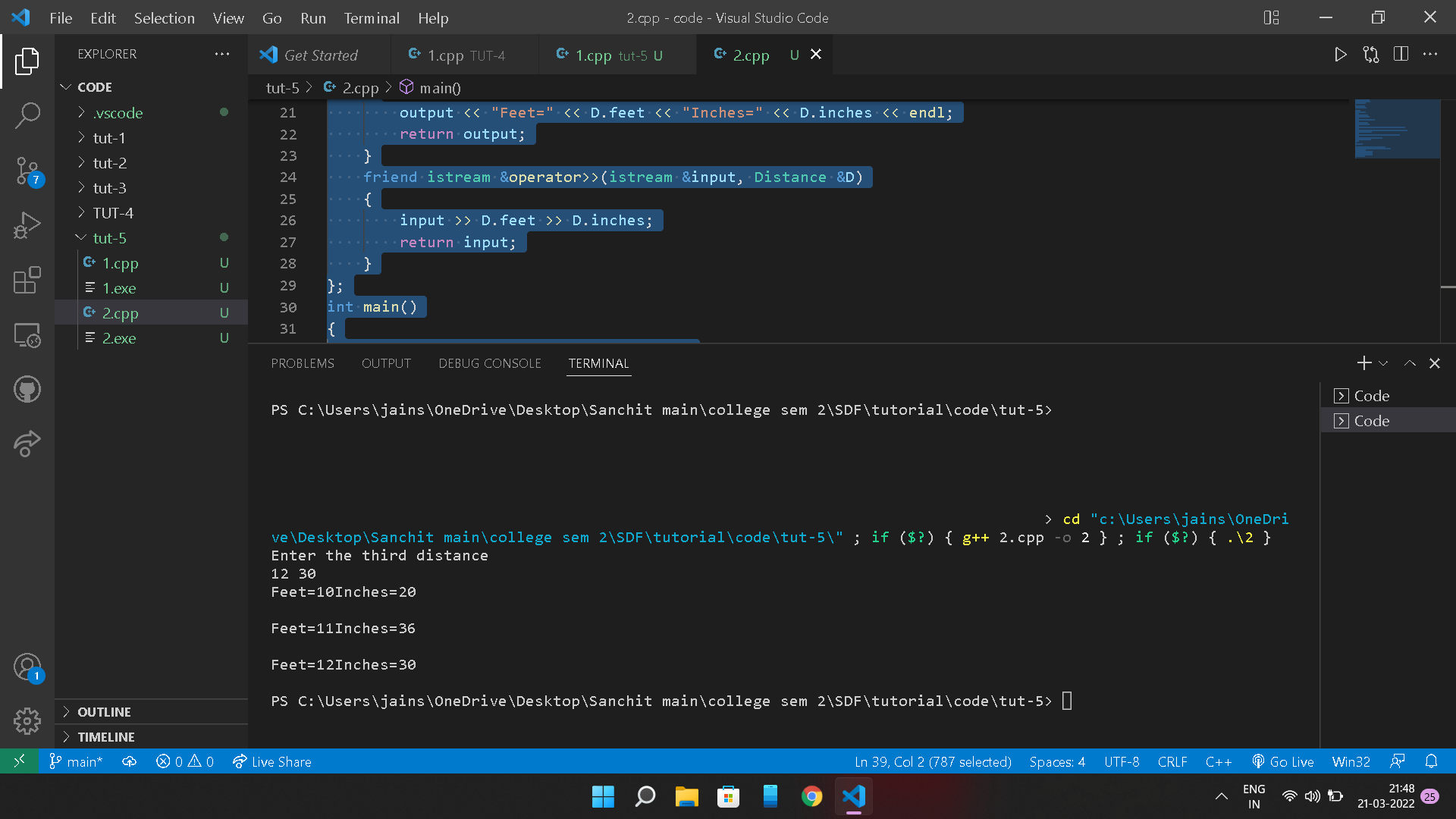
    cout << D1 << endl;

    cout << D2 << endl;

    cout << D3 << endl;

    return 0;

}



**2.**

#include <iostream>

using namespace std;

class Distance

{

    int feet;

    int inches;

public:

    Distance()

    {

        feet = 0;

        inches = 0;

    }

    Distance(int f, int i)

    {

        feet = f;

        inches = i;

    }

    bool operator<(const Distance &D2)

    {

        if (feet < D2.feet)

        {

            return true;

        }

        if (feet == D2.feet && inches < D2.inches)

        {

            return true;

        }

        else

        {

            return false;

        }

    }

};

int main()

{

    Distance D1(10, 20);

    Distance D2(20, 40);

    if (D1 < D2)

    {

        cout << "D1 is smaller" << endl;

    }

    else

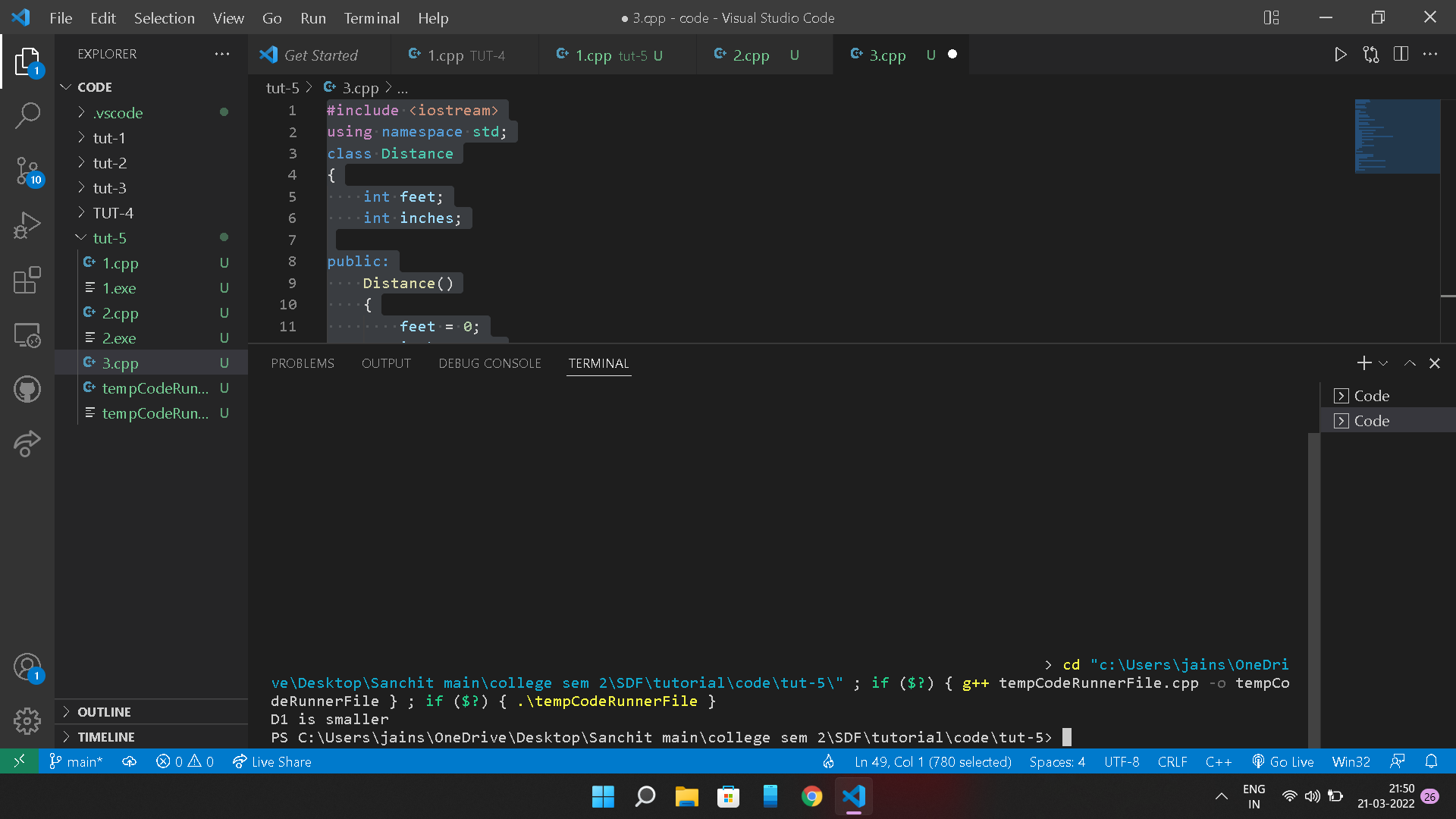
    {

        cout << "D2 is smaller" << endl;

    }

    return 0;

}



**2.** #include <iostream>

using namespace std;

class Distance

{

    int feet;

    int inches;

public:

    Distance()

    {

        feet = 0;

        inches = 0;

    }

    Distance(int f, int i)

    {

        feet = f;

        inches = i;

    }

    Distance operator()(int a, int b, int c)

    {

        Distance D;

        D.feet = a + c + 10;

        D.inches = b + c + 100;

        return D;

    }

    void displayDistance()

    {

        cout << "F:" << feet << endl;

        cout << "I:" << inches << endl;

    }

};

int main()

{

    Distance D1(10, 11), D2;

    cout << "First distance:";

    D1.displayDistance();

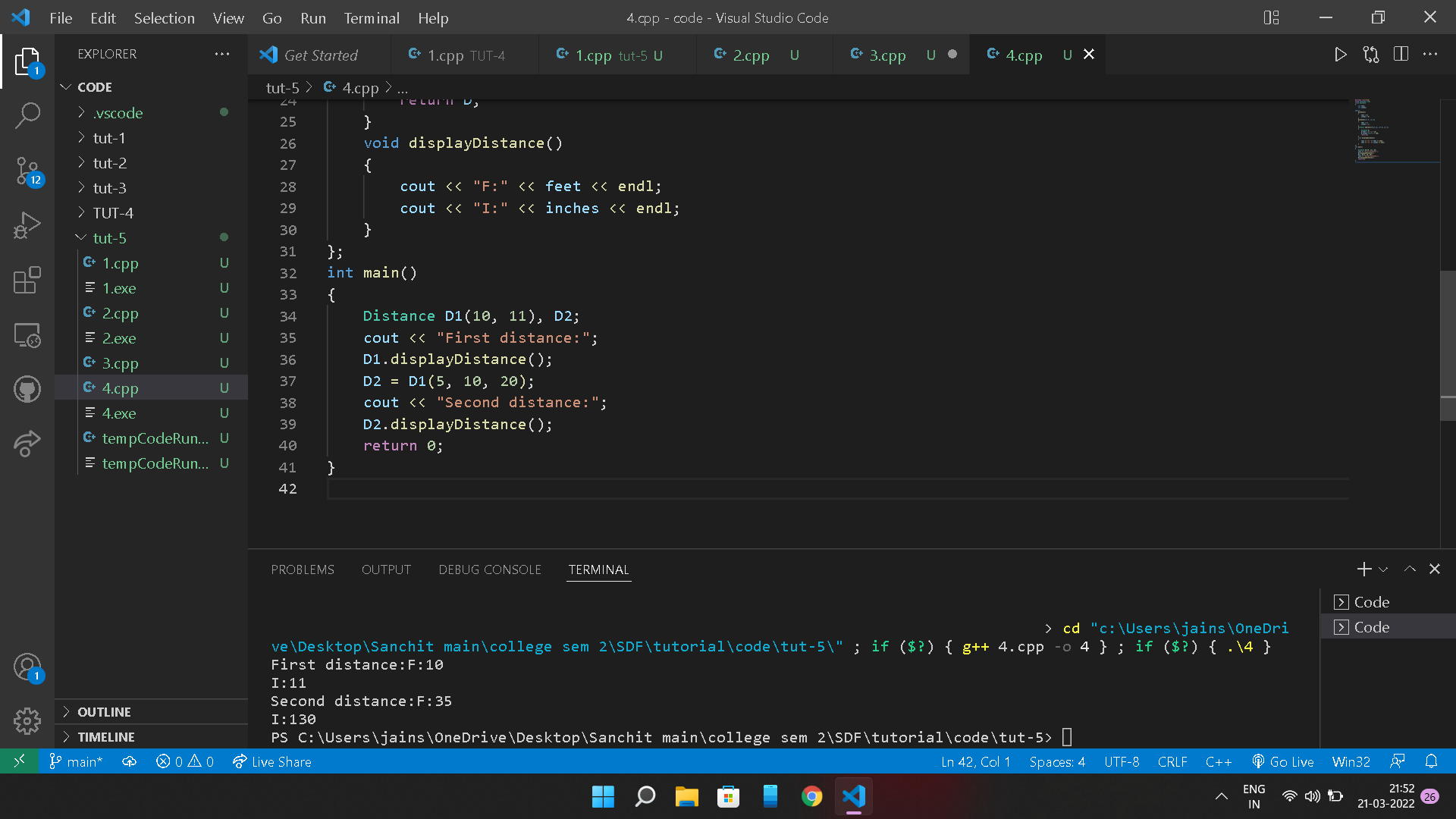
    D2 = D1(5, 10, 20);

    cout << "Second distance:";

    D2.displayDistance();

    return 0;

}



**3.** #include <iostream>

#include <string.h>

using namespace std;

class mystring

{

    char str[50];

public:

    string operator!();

    void inp\_st()

    {

        cout << "Enter the string" << endl;

        cin >> str;

    }

    void display()

    {

        cout << str << endl;

    }

};

string mystring::operator!()

{

    for (int i = 0; str[i] != '\_'; i++)

    {

        if (str[i] >= 65 && str[i] <= 96)

        {

            str[i] = str[i] + 32;

        }

        else if (str[i] >= 97 && str[i] <= 122)

        {

            str[i] = str[i] - 32;

        }

    }

    cout << "Reversed string is" << str << endl;

}

int main()

{

    mystring s;

    s.inp\_st();

    s.display();

    !s;

    // cout<<"Reversed string is:"<<endl;

    // s.display();

    return 0;

}