***NAME : Himanshu Dixit***

***ENROLL NO. : 21103262***

***BATCH : B10***

***Software Development fundamentals-2 [EVEN 2022]***

***Tutorial Sheet -5 (Week 5)***

***Q1.*** *What will be the output of the following C++ code?*

***(i)***

*#include <iostream>*

*#include <string>*

*using namespace std;*

*class A*

*{*

*static int a;*

*public:*

*void show()*

*{*

*a++;*

*cout<<"a: "<<a<<endl;*

*}*

*};*

*int A::a = 5;*

*int main(int argc, char const \*argv[])*

*{*

*A a;*

*return 0;*

*}*

***Solution :***

*It will print nothing*

***Explanation :***

*As every static member must be initialized and we have initialized variable*

*‘a’ so no runtime error.Also as variable ‘a’ is a static member and is referenced using The class for initialization therefore no compiler error.*

***(ii)***

*#include <iostream>*

*using namespace std;*

*class Count {*

*private:*

*int value;*

*public:*

*// Constructor to initialize count to 5*

*Count() : value(5) {}*

*// Overload ++ when used as prefix*

*void operator ++ () {*

*++value;*

*}*

*// Overload ++ when used as postfix*

*void operator ++ (int) {*

*++value;*

*}*

*void display() {*

*cout << "Count: " << value << endl;*

*}*

*};*

*int main() {*

*Count count1;*

*// Call the "void operator ++ (int)" function*

*count1++;*

*count1.display();*

*// Call the "void operator ++ ()" function*

*++ count1;*

*count1.display();*

*return 0;*

*}*

***Solution :***

*Count: 6*

*Count: 7*

***(iii)***

*#include<stdlib.h>*

*#include<stdio.h>*

*#include<iostream>*

*using namespace std;*

*class Test {*

*int x;*

*public:*

*void\* operator new(size\_t size);*

*void operator delete(void\*);*

*Test(int i) {*

*x = i;*

*cout << "Constructor called n"; }*

*~Test() { cout << "Destructor called n"; }*

*};*

*void\* Test::operator new(size\_t size)*

*{*

*void \*storage = malloc(size);*

*cout << "new called n";*

*return storage;*

*}*

*void Test::operator delete(void \*p )*

*{*

*cout<<"delete called n";*

*free(p);*

*}*

*int main()*

*{*

*Test \*m = new Test(5);*

*delete m;*

*return 0;*

*}*

***Solution :***

*new called*

*Constructor called*

*Destructor called*

*delete called*

***Explanation:***

*Consider the following statement*

*Test\*ptr=newTest;*

*There are actually 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)***

*#include<iostream>*

*using namespace std;*

*class A*

*{*

*int i;*

*public:*

*A(int ii = 0) : i(ii) {}*

*void show() { cout << i << endl; }*

*};*

*class B*

*{*

*int x;*

*public:*

*B(int xx) : x(xx) {}*

*operator A() const { return A(x); }*

*};*

*void g(A a)*

*{*

*a.show();*

*}*

*int main()*

*{*

*B b(10);*

*g(b);*

*g(20);*

*return 0;*

*}*

***Solution :***

*10*

*20*

***Explanation:***

*Note that the 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)***

*#include<iostream>*

*using namespace std;*

*class Point {*

*private:*

*int x, y;*

*public:*

*Point() : x(0), y(0) { }*

*Point& operator()(int dx, int dy);*

*void show() {cout << "x = " << x << ", y = " << y; }*

*};*

*Point& Point::operator()(int dx, int dy)*

*{*

*x = dx;*

*y = dy;*

*return \*this;*

*}*

*int main()*

*{*

*Point pt;*

*pt(3, 2);*

*pt.show();*

*return 0;*

*}*

***Solution :***

*x = 3, y = 2*

***Explanation:***

*This a simple example of function call operator overloading. The function call operator, when overloaded, does not modify how functions are called. Rather, it modifies how the operator is To be interpreted when applied to objects of a given type. If you overload a function call Operator for a class its declaration will have the following form:*

*return\_typeoperator()(parameter\_list*

***Q2.*** *Write C++ program to overload the following operators-*

***(i)*** *==*

***(ii)*** *>>, <<*

***(iii****) relational (>, <, …)*

***(iv)*** *function call*

***Solution :***

***(i)***

*#include <iostream>*

*using namespace std;*

*class Time*

*{*

*int hr,min,sec;*

*public:*

*//default constructor*

*Time()*

*{*

*hr=0,min=0;sec=0;*

*}*

*//overloaded constructor*

*Time(int h,int m,int s)*

*{*

*hr=h;*

*min=m;*

*sec=s;*

*}*

*//overloading '==' operator*

*friend bool operator==(Time &t1,Time &t2);*

*};*

*bool operator==(Time&t1,Time&t2)*

*{*

*return(t1.hr==t2.hr && t1.min==t2.min && t1.sec==t2.sec);*

*}*

*int main()*

*{*

*Time t1(3,15,45);*

*Time t2(4,15,45);*

*if(t1==t2)*

*{*

*cout<<"Both the time values are equal";*

*}*

*else*

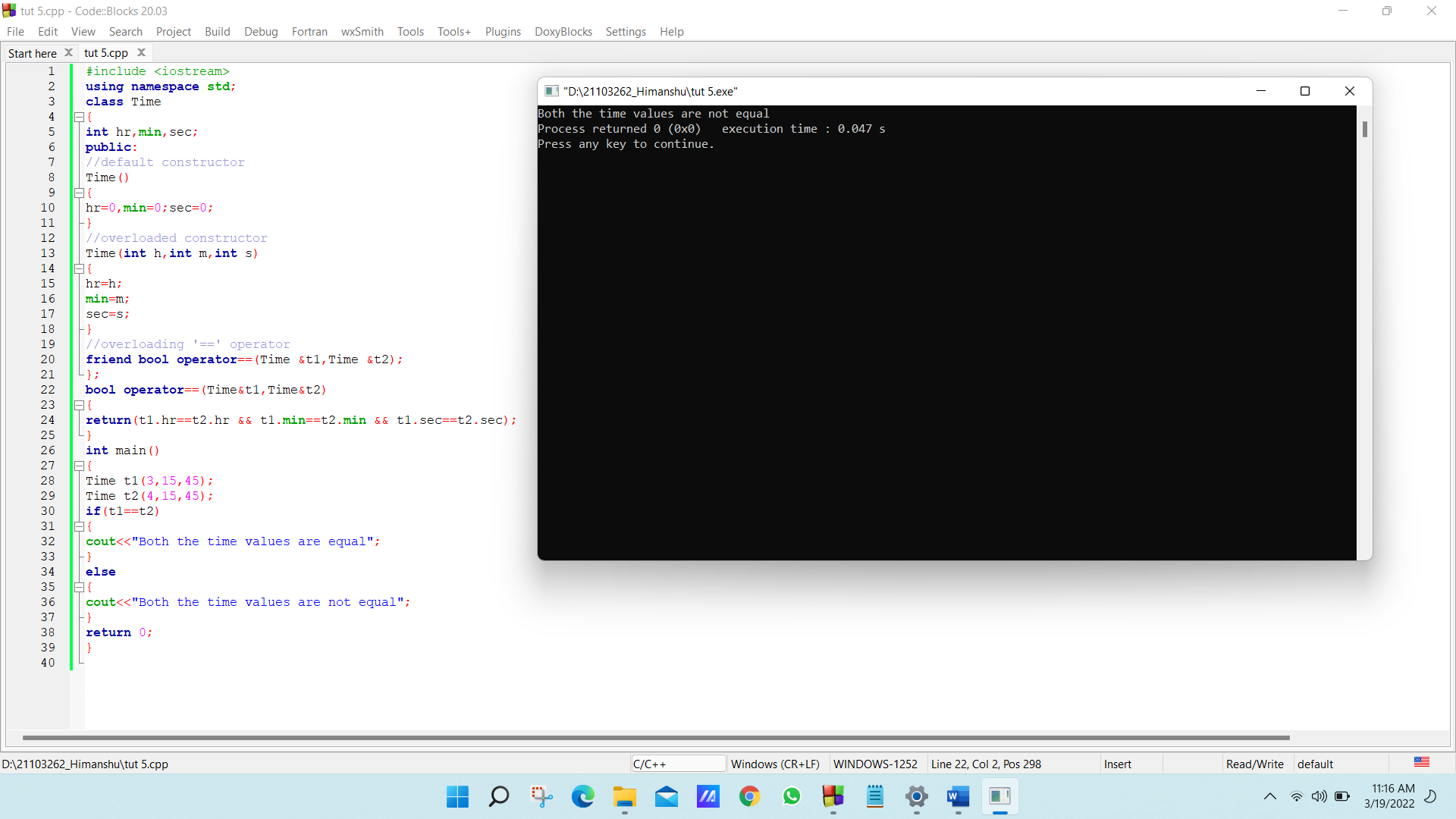
*{*

*cout<<"Both the time values are not equal";*

*}*

*return 0;*

*}*



***(ii)***

*#include <iostream>*

*using namespace std;*

*class Distance{*

*private:*

*int feet; //0 to infinite*

*int inches; //0 to 12*

*public:*

*//required constructors*

*Distance(){*

*feet=0;*

*inches=0;*

*}*

*Distance(int f,int i){*

*feet=f;*

*inches=i;*

*}*

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

*{*

*output<<"F:"<<D.feet<<"I:"<<D.inches;*

*return output;*

*}*

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

*{*

*input>>D.feet>>D.inches;*

*return input;*

*}*

*};*

*int main(){*

*Distance D1(11,10),D2(5,11),D3;*

*cout<<"Enter the value of object:"<<endl;*

*cin>>D3;*

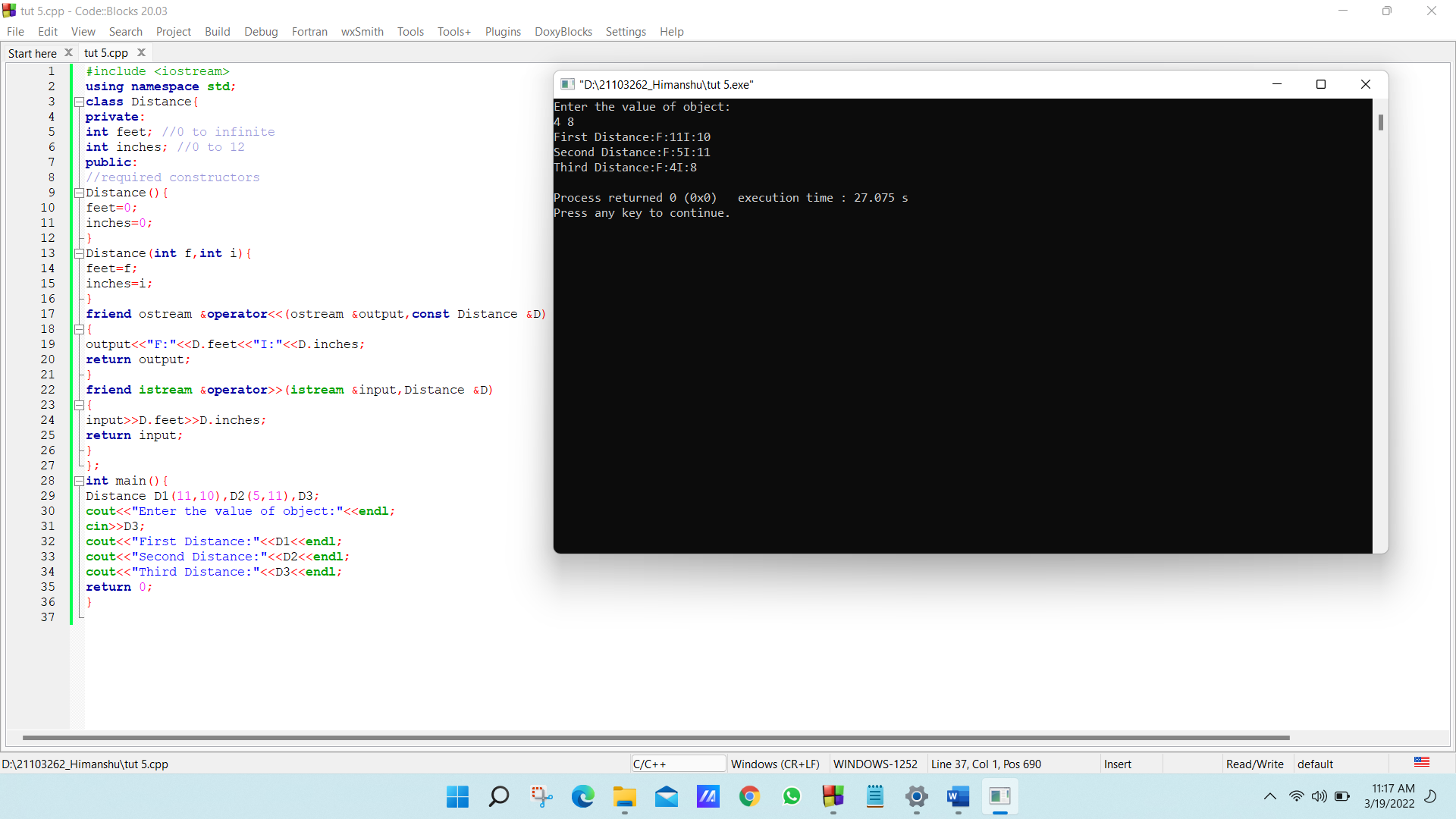
*cout<<"First Distance:"<<D1<<endl;*

*cout<<"Second Distance:"<<D2<<endl;*

*cout<<"Third Distance:"<<D3<<endl;*

*return 0;*

*}*



***(iii)***

*#include <iostream>*

*Using namespace std;*

*class Distance{*

*private:*

*int feet; //0 to infinite*

*int inches; //0 to 12*

*public:*

*//required constructors*

*Distance(){*

*feet=0;*

*inches=0;*

*}*

*Distance(int f,int i){*

*feet=f;*

*inches=i;*

*}*

*//overloaded minus(-)operator*

*Distance operator-(){*

*feet=-feet;*

*inches=-inches;*

*return Distance (feet,inches);*

*}*

*//overloaded <operator*

*Bool operator<(const Distance &d)*

*{*

*if(feet<d.feet){*

*return true;*

*}*

*if(feet==d.feet && inches<d.inches)*

*{*

*return true;*

*}*

*return false;*

*}*

*};*

*int main(){*

*Distance D1(11,10),D2(5,11);*

*if(D1<D2){*

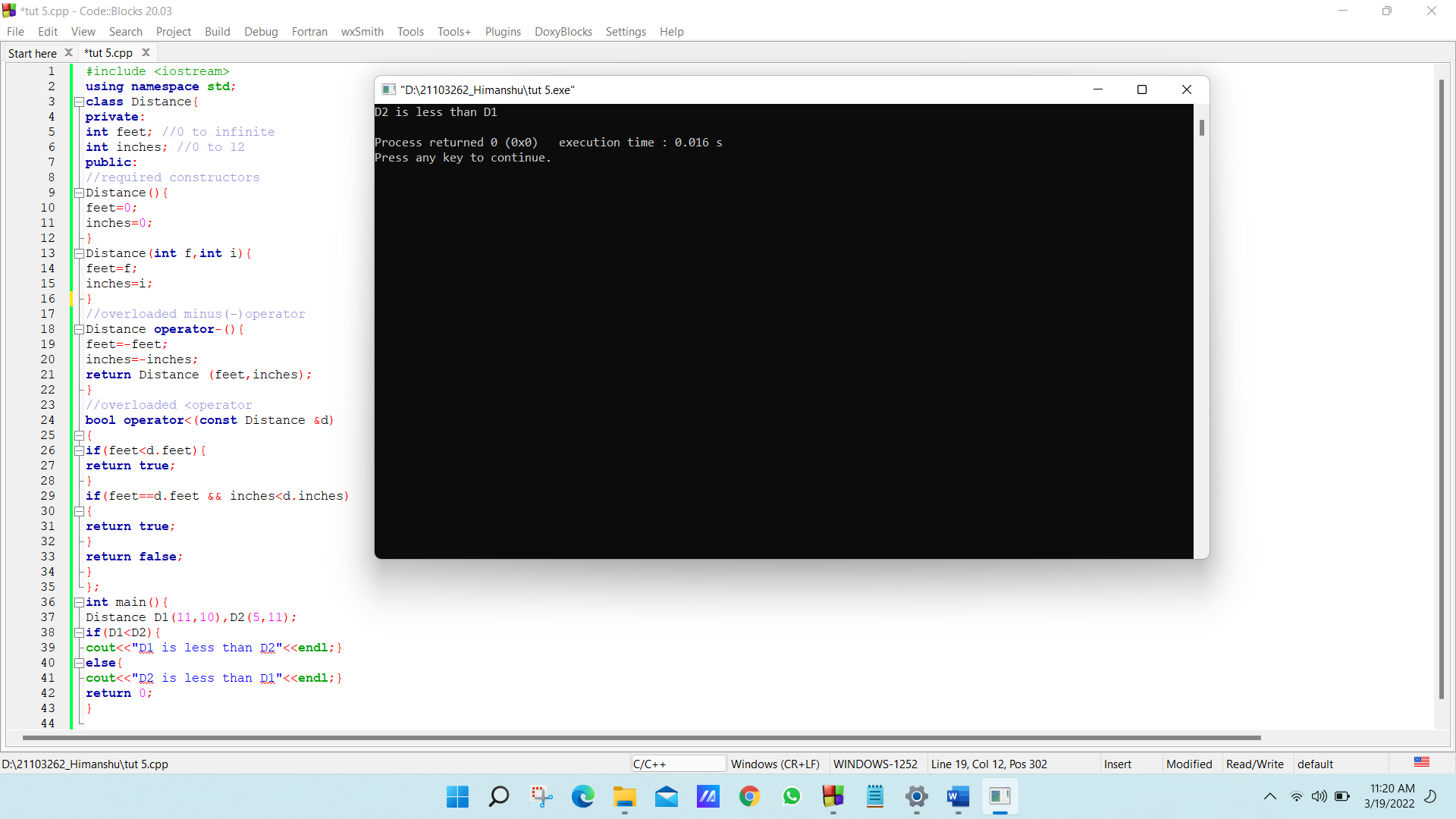
*cout<<"D1 is less than D2"<<endl;}*

*else{*

*cout<<"D2 is less than D1"<<endl;}*

*return 0;*

*}*



***(iv)***

*#include <iostream>*

*using namespace std;*

*class Distance{*

*private:*

*int feet; // 0 to infinite*

*int inches; //0 to 12*

*public:*

*//required constructors*

*Distance(){*

*feet=0;*

*inches=0;*

*}*

*Distance (int f,int i){*

*feet=f;*

*inches=i;*

*}*

*//overload function call*

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

*{*

*Distance D;*

*//just put random calculation*

*D.feet=a+c+10;*

*D.inches=b+c+100;*

*return D;*

*}*

*//method to display distance*

*void display Distance(){*

*cout<<"F:"<<feet<<"I:"<<inches<<endl;*

*}*

*};*

*int main(){*

*Distance D1(11,10),D2;*

*cout<<"First Distance:";*

*D1.displayDistance();*

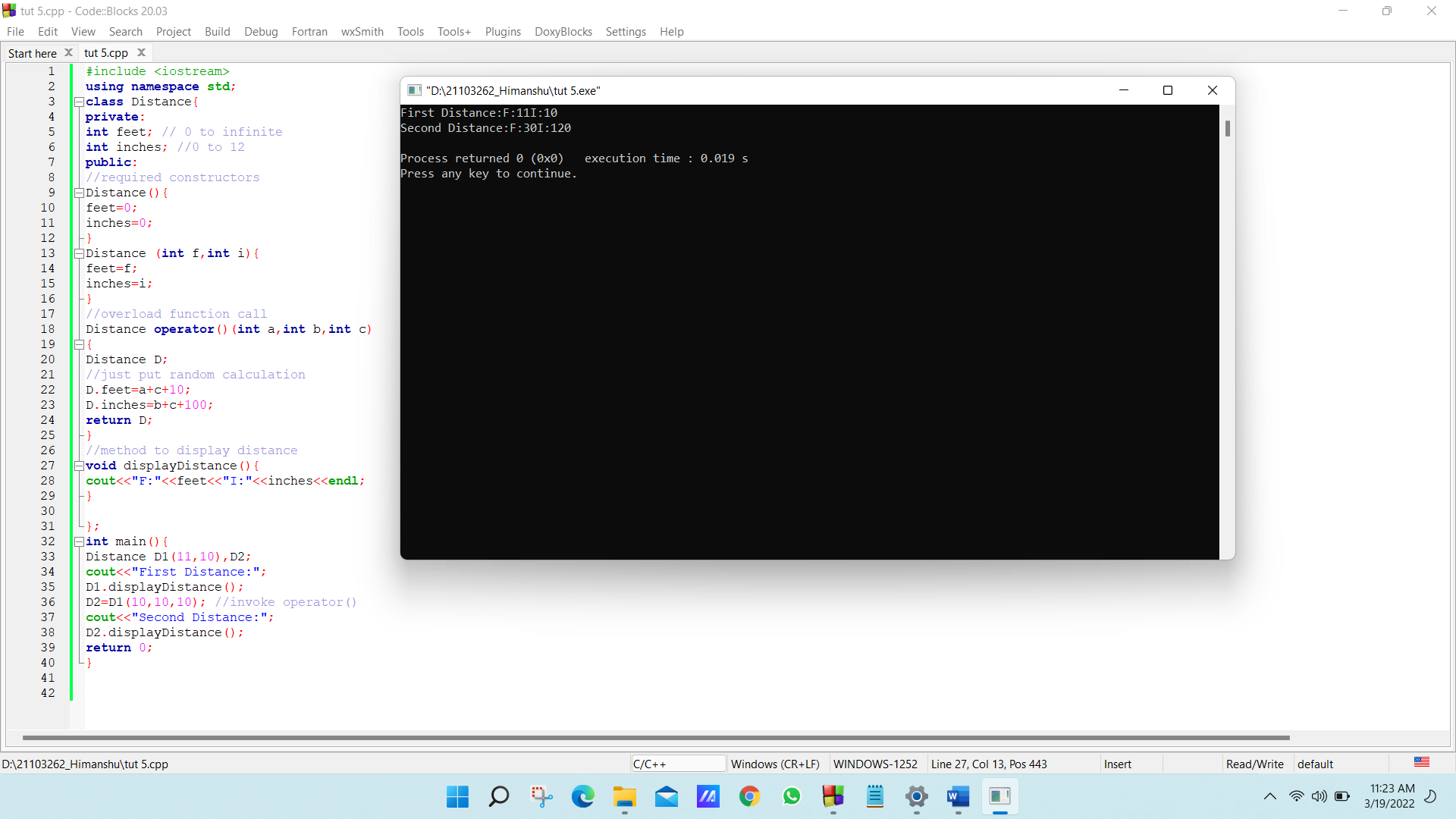
*D2=D1(10,10,10); //invoke operator()*

*cout<<"Second Distance:";*

*D2.displayDistance();*

*return 0;*

*}*



***Q3.*** *WAP in C++ to reverse the case of alphabets using Operator overloading.*

***Solution :***

*#include <iostream>*

*#include <string>*

*using namespace std;*

*class mystring*

*{*

*char str[500];*

*public:*

*void operator!(); //Overloaded'!'Operator*

*void accept\_string()*

*{*

*cout<<"\nEnterString:";*

*cin>>str;*

*}*

*void display\_string()*

*{*

*cout<<str;*

*}*

*};*

*void 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<<"\n\nReverse Case String is:"<<str;*

*}*

*int main()*

*{*

*mystring s1;*

*s1.accept\_string();*

*cout<<"\n\nString is:";*

*s1.display\_string();*

*!s1;*

*return 0;*

*}*

