**A**

**Lab File/ Report**

Of

**OOPS Lab**

**LC-CSE-214G**

[In the partial fulfilment of Four Year BTech (Computer Science & Engg.) course]

**Submitted To: Submitted By:**

**Mrs. Rashmi Verma Neonika Sehrawat**

**HOD – CSE Stu ID: 91237**

**Dept. Of Computer Science BTech (CSE) – 4th Sem**

****

**2021**

**Department of Computer Science & Engineering**

**DPG Institute of Technology & Management, Gurgaon (HR)**

**Maharishi Dayanand University, Rohtak (HR)**

**INDEX**

1. *Write a program that uses a class where the member functions are defined inside a class.*
2. *Write a program that uses a class where the member functions are defined outside a class.*
3. *Write a program to demonstrate the use of static data members.*
4. *Write a program to demonstrate the use of constant data members.*
5. *Write a program to demonstrate the use of zero argument and parameterized constructors.*
6. *Write a program to demonstrate the use of dynamic constructor.*
7. *Write a program to demonstrate the use of explicit constructor.*
8. *Write a program to demonstrate the use of initializer list.*
9. *Write a program to demonstrate the overloading of increment and decrement operators.*
10. *Write a program to demonstrate the overloading of binary arithmetic operators.*
11. *Write a program to demonstrate the overloading of memory management operators*
12. *Write a program to demonstrate the multilevel inheritance.*
13. *Write a program to demonstrate the multiple inheritance*
14. *Write a program to demonstrate the virtual derivation of a class.*
15. *Write a program to demonstrate the runtime polymorphism.*
16. *Write a program to demonstrate the exception handling.*
17. *Write a program to demonstrate the use of function template.*
18. *Write a program to demonstrate the use of class template.*

**PRACTICALS**

1. *Write a program that uses a class where the member functions are defined inside a class.*

#include <iostream>

using namespace std;

class Employee

{

int id;

public:

void setData(void)

{

cout<<"Enter the id"<<endl;

cin>>id;

}

void getData(void)

{

cout<<"The id of this employee is "<<id<<endl;

}

};

int main(){

Employee Phoebe, Monica, Rachel;

Phoebe.setData();

Phoebe.getData();

Monica.setData();

Monica.getData();

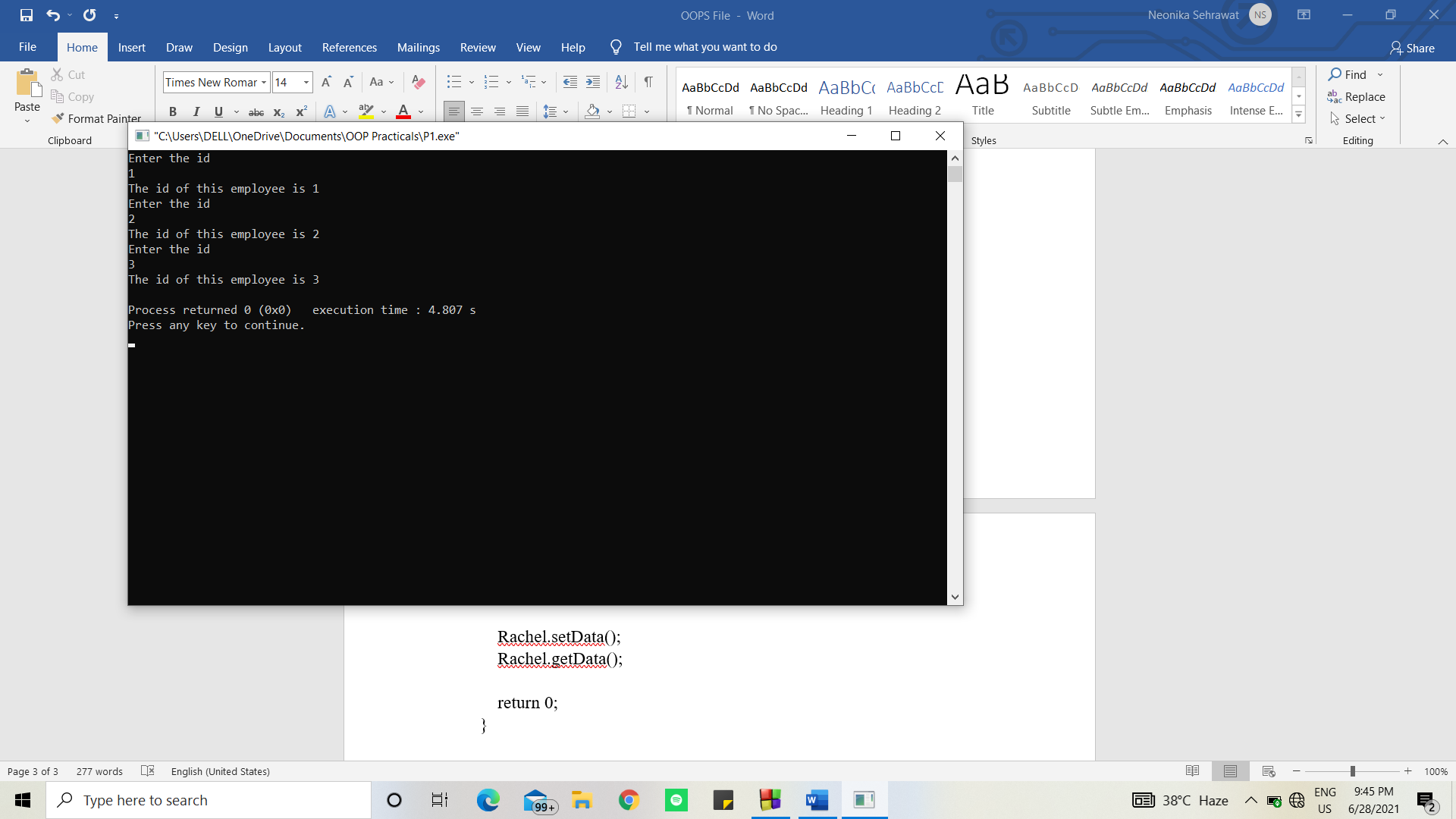
Rachel.setData();

Rachel.getData();

return 0;

}

Output:



1. *Write a program that uses a class where the member functions are defined outside a class.*

#include<iostream>

using namespace std;

class Shop

{

int itemId;

int itemPrice;

public:

void setPrice(void);

void displayPrice(void);

};

void Shop::setPrice(void)

{

cout<<"Enter Id of your item no. "<<endl;

cin>>itemId;

cout<<"Enter Price of your item"<<endl;

cin>>itemPrice;

}

void Shop::displayPrice(void)

{

cout<<"The Price of item with Id "<<itemId<<" is "<<itemPrice<<endl;

}

int main(){

Shop store1, store2, store3;

store1.setPrice();

store1.displayPrice();

store2.setPrice();

store2.displayPrice();

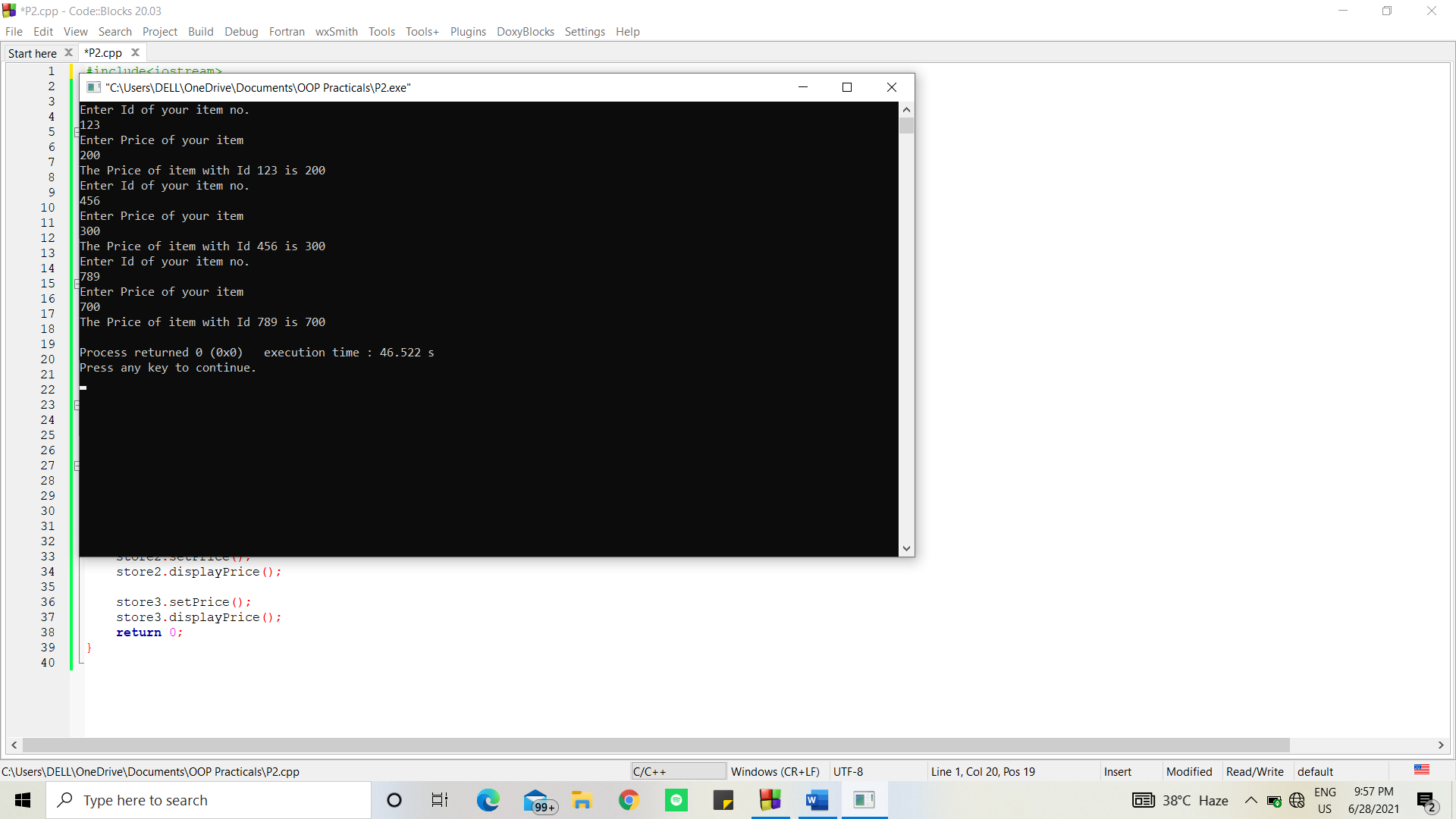
store3.setPrice();

store3.displayPrice();

return 0;

}

Output:



1. *Write a program to demonstrate the use of static data members.*

#include <iostream>

using namespace std;

class Employee

{

int id;

static int count;

public:

void setData(void)

{

cout<<"Enter the id"<<endl;

cin>>id;

count++;

}

void getData(void)

{

cout<<"The id of this employee is "<<id<<endl;

}

static void getCount(void){

cout<<"The value of count is "<<count<<endl;

}

};

//count is the static data member of class Employee

int Employee::count; //default value is 0

int main(){

Employee Phoebe, Monica, Rachel;

Phoebe.setData();

Phoebe.getData();

Employee::getCount();

Monica.setData();

Monica.getData();

Employee::getCount();

Rachel.setData();

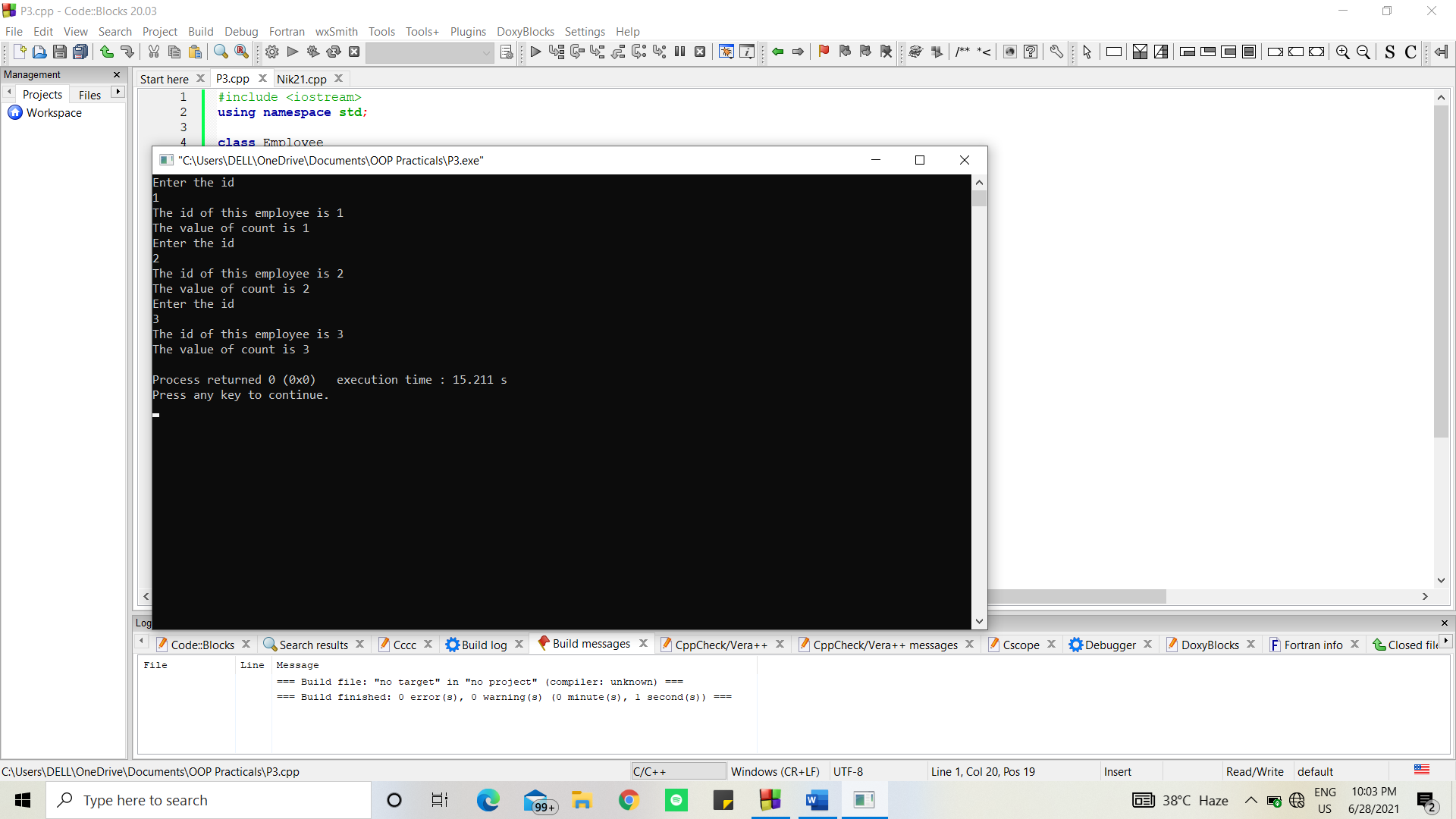
Rachel.getData();

Employee::getCount();

return 0;

}

Output:



1. *Write a program to demonstrate the use of constant data members.*

#include<iostream>

using namespace std;

class Demo {

int val;

public:

Demo(int x = 0) {

val = x;

}

int getValue() const {

return val;

}

};

int main() {

const Demo d(96);

Demo d1(13);

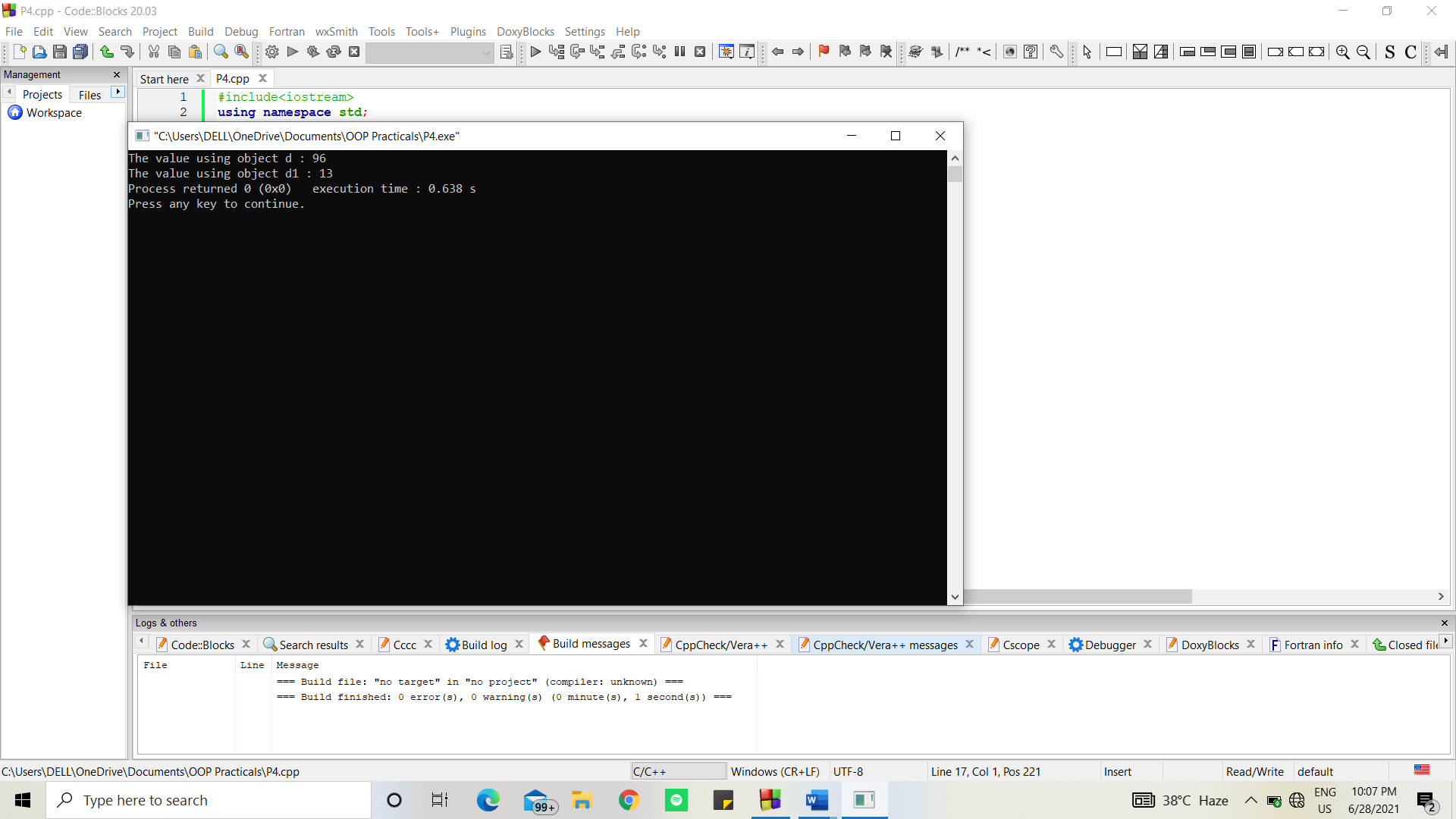
cout << "The value using object d : " << d.getValue();

cout << "\nThe value using object d1 : " << d1.getValue();

return 0;

}

Output:



1. *Write a program to demonstrate the use of zero argument and parameterized constructors.*

#include<iostream>

using namespace std;

class Complex

{

int a, b;

public:

Complex(int, int); // Constructor declaration

void printNumber()

{

cout << "Your number is " << a << " + " << b << "i" << endl;

}

};

Complex ::Complex(int x, int y) // ----> This is a parameterized constructor as it takes 2 parameters

{

a = x;

b = y;

}

int main(){

// Implicit call

Complex a(4, 6);

a.printNumber();

// Explicit call

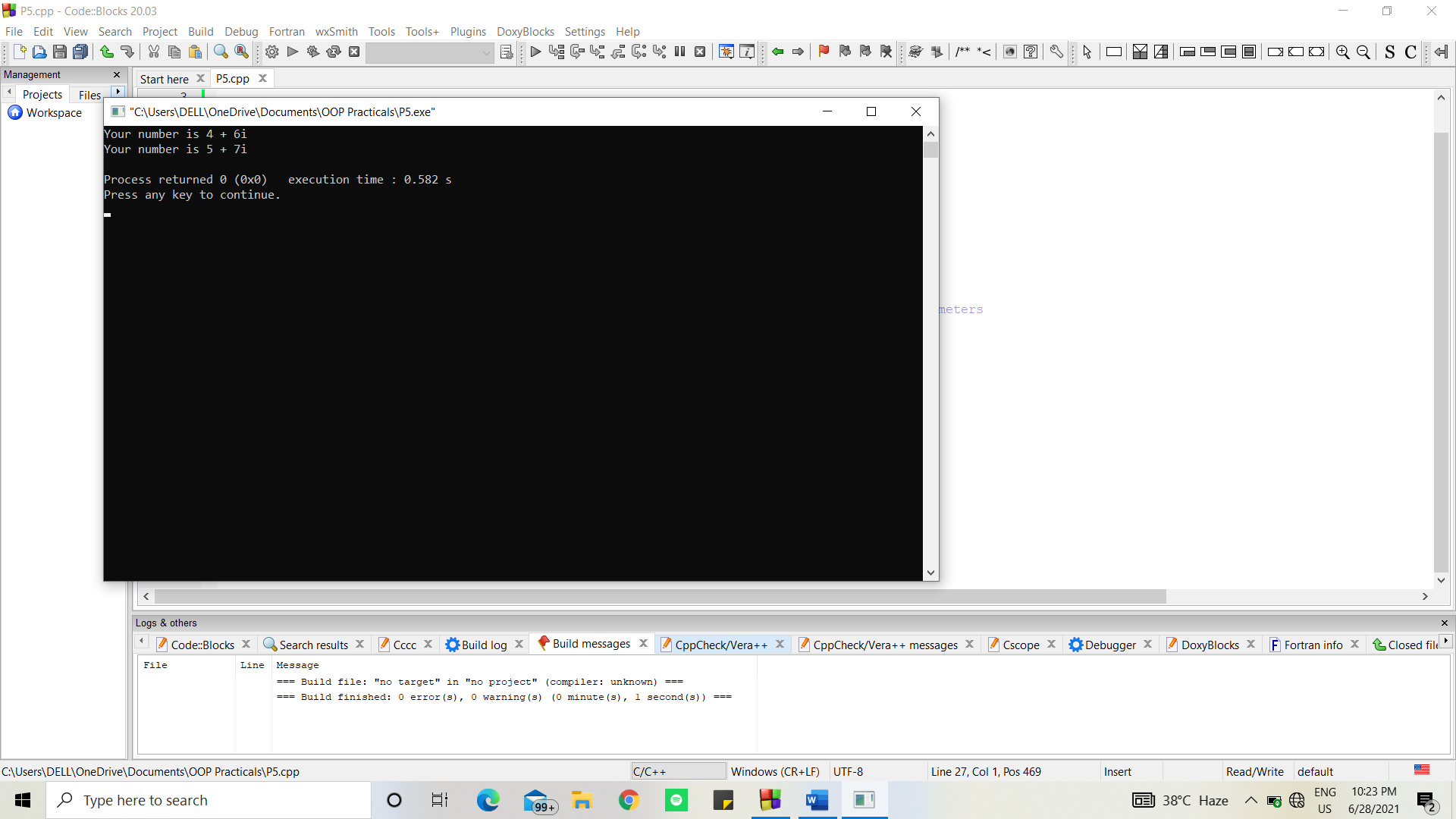
Complex b = Complex(5, 7);

b.printNumber();

return 0;

}

Output:



1. *Write a program to demonstrate the use of dynamic constructor.*

#include<iostream>

using namespace std;

class BankDeposit{

int principal;

int years;

float interestRate;

float returnValue;

public:

BankDeposit(){}

BankDeposit(int p, int y, float r); // r can be a value like 0.04

BankDeposit(int p, int y, int r); // r can be a value like 14

void show();

};

BankDeposit :: BankDeposit(int p, int y, float r)

{

principal = p;

years = y;

interestRate = r;

returnValue = principal;

for (int i = 0; i < y; i++)

{

returnValue = returnValue \* (1+interestRate);

}

}

BankDeposit :: BankDeposit(int p, int y, int r)

{

principal = p;

years = y;

interestRate = float(r)/100;

returnValue = principal;

for (int i = 0; i < y; i++)

{

returnValue = returnValue \* (1+interestRate);

}

}

void BankDeposit :: show(){

cout<<endl<<"Principal amount was "<<principal

<< ". Return value after "<<years

<< " years is "<<returnValue<<endl;

}

int main(){

BankDeposit bd1, bd2, bd3;

int p, y;

float r;

int R;

cout<<"Enter the value of p y and r"<<endl;

cin>>p>>y>>r;

bd1 = BankDeposit(p, y, r);

bd1.show();

cout<<"Enter the value of p y and R"<<endl;

cin>>p>>y>>R;

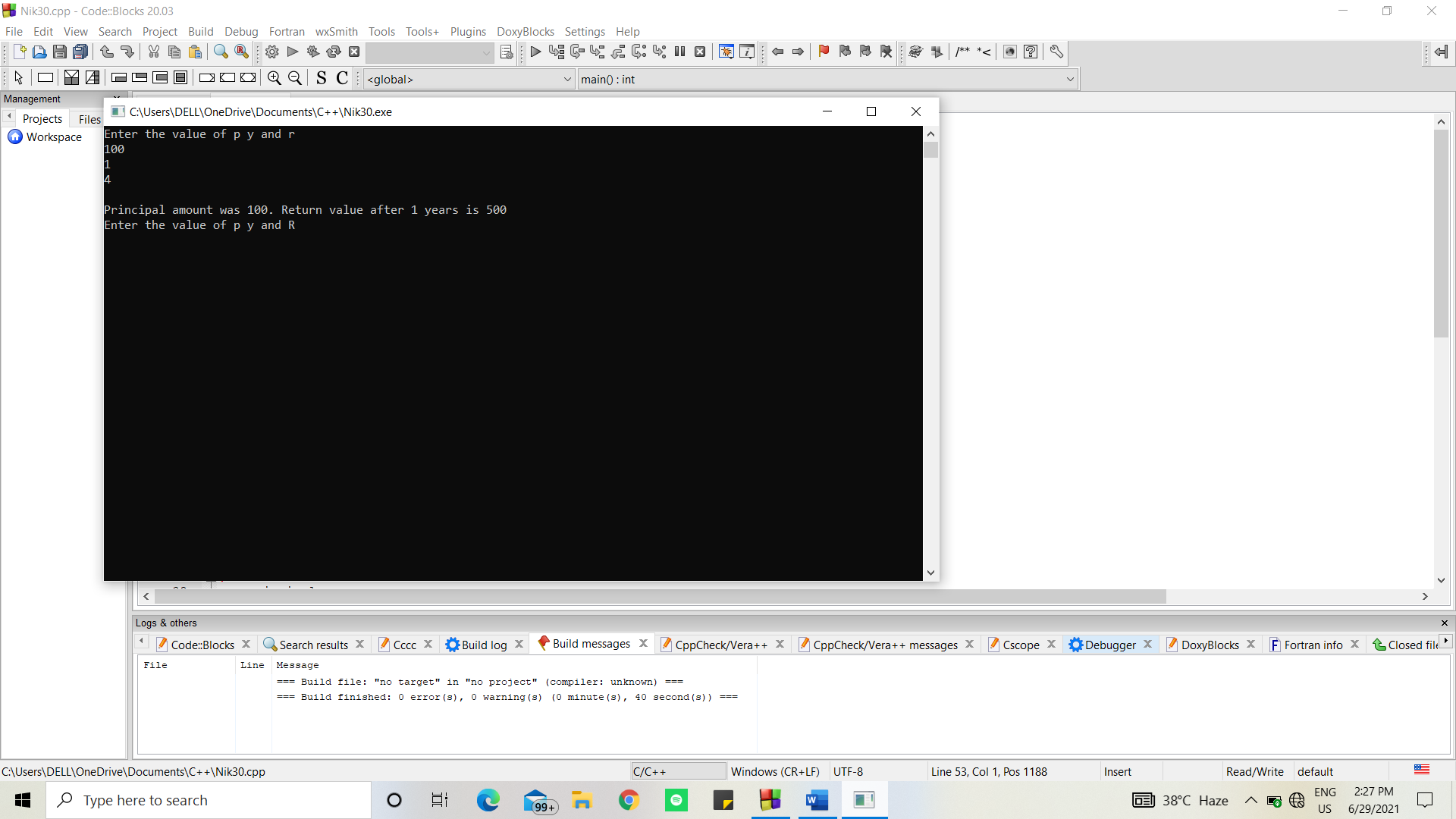
bd2 = BankDeposit(p, y, R);

bd2.show();

return 0;

}

Output:



1. *Write a program to demonstrate the use of explicit constructor.*

#include<iostream>

using namespace std;

//Explicit constructor avoids implicit call to the constructor

class Base{

int var;

public:

Base(){}

explicit Base(int var2) : var(var2) {}

void print() {

cout<<var<<endl;

}

};

void func(Base b) {

b.print();

}

int main(){

Base obj1(13); //Normal call to constructor

Base obj2 = 24; //Implicit call to constructor

func(obj1); //Normal call to constructor

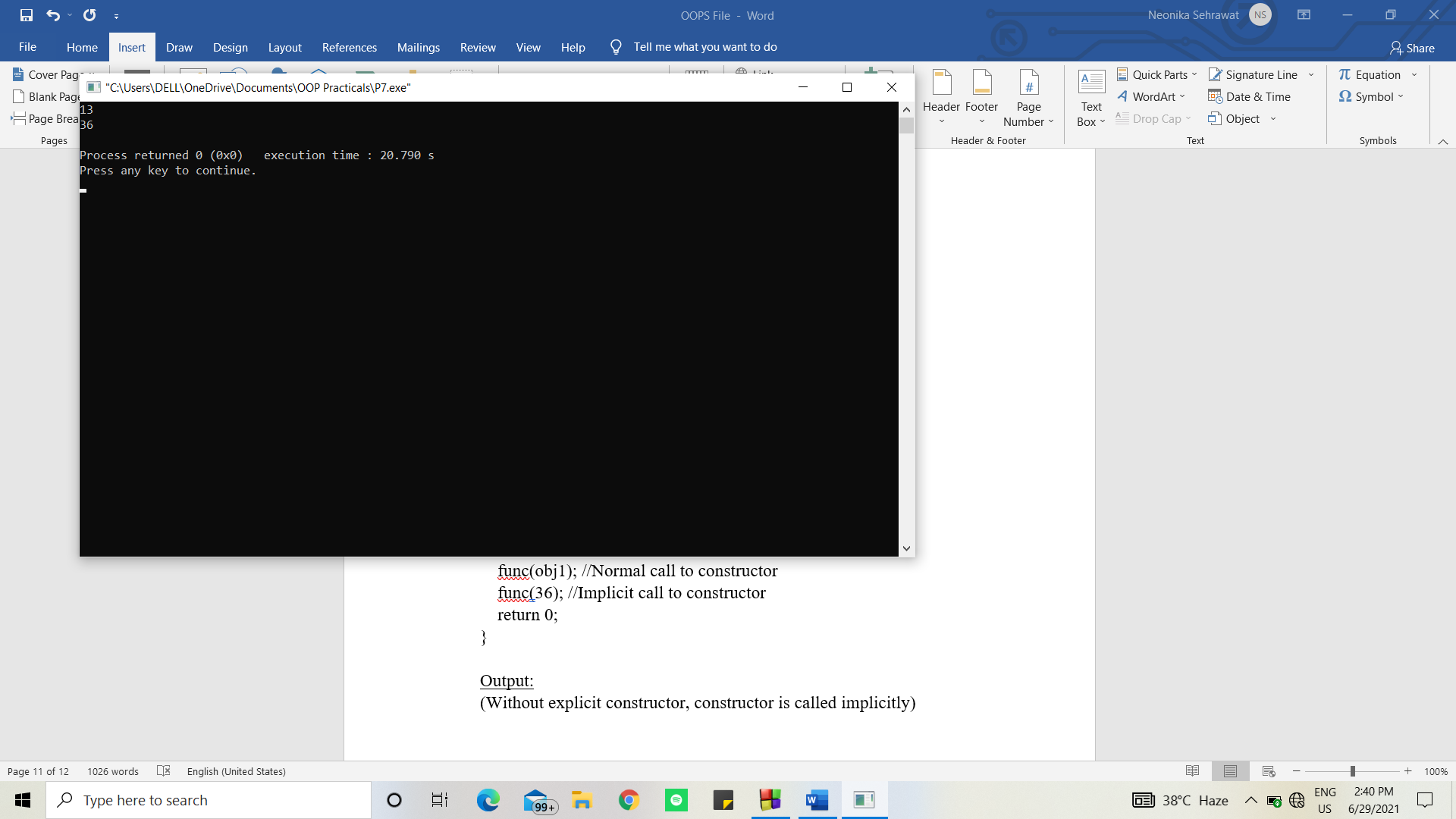
func(36); //Implicit call to constructor

return 0;

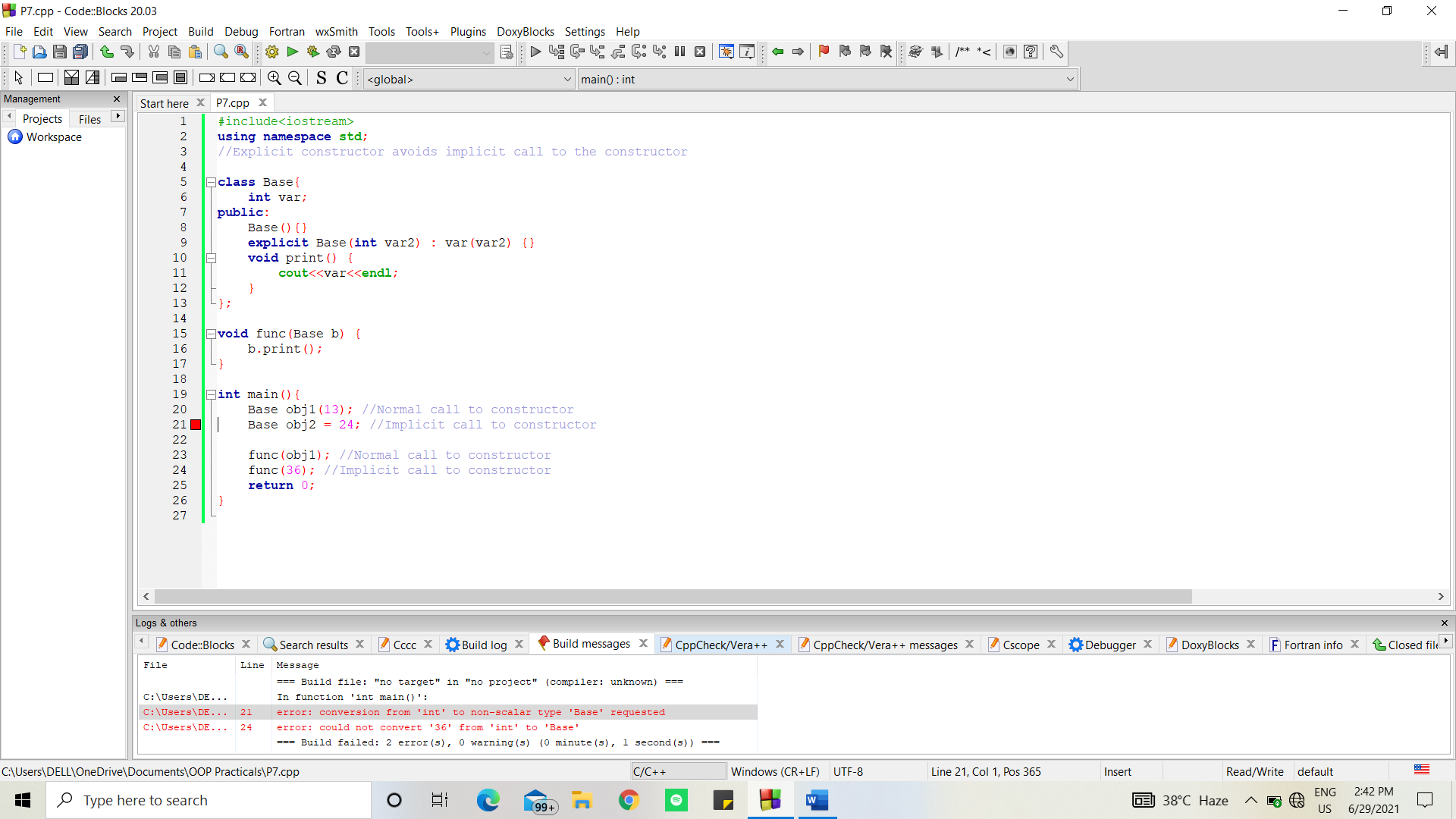
}

Output:

(Without explicit constructor, constructor is called implicitly)



(With Explicit constructor error is shown)



1. *Write a program to demonstrate the use of initializer list.*

#include <iostream>

using namespace std;

class Test

{

int a;

int b;

public:

Test(int i, int j)

{

a = i;

b = j;

cout << "Value of a is "<<a<<endl;

cout << "Value of b is "<<b<<endl;

}

};

int main()

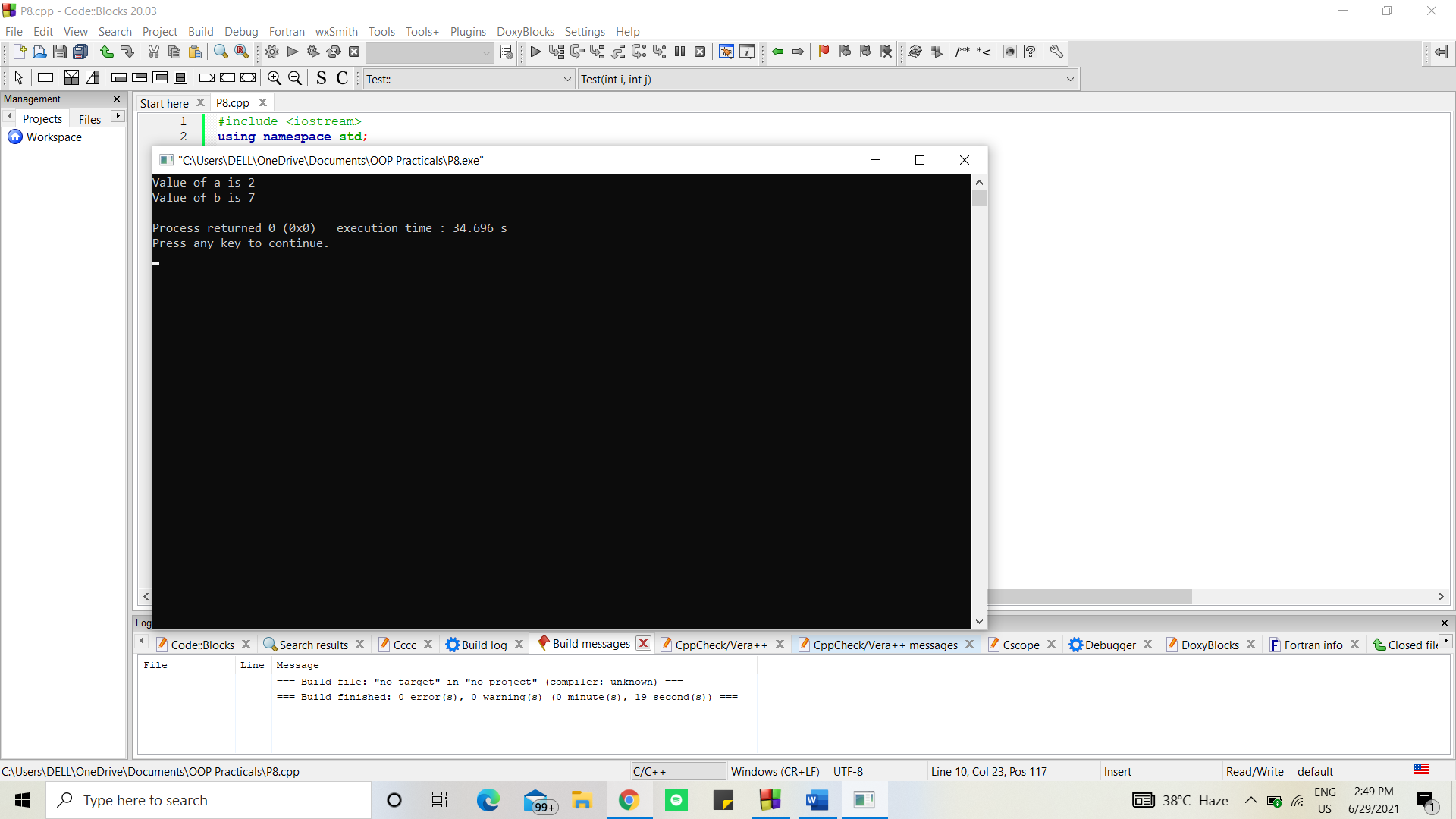
{

Test t(2, 7);

return 0;

}

Output:



1. *Write a program to demonstrate the overloading of increment and decrement operators.*

#include<iostream>

using namespace std;

/\*

++a --> Increment by 1 and then use the value of a

a++ --> Use the value of a and the increment by 1

--a --> Decrement by 1 and then use the value of a

a-- --> Use the value of a and the decrement by 1

\*/

int main(){

int a = 13;

cout<<a++<<endl;

cout<<a<<endl;

cout<<++a<<endl;

cout<<a<<endl;

cout<<--a<<endl;

cout<<a<<endl;

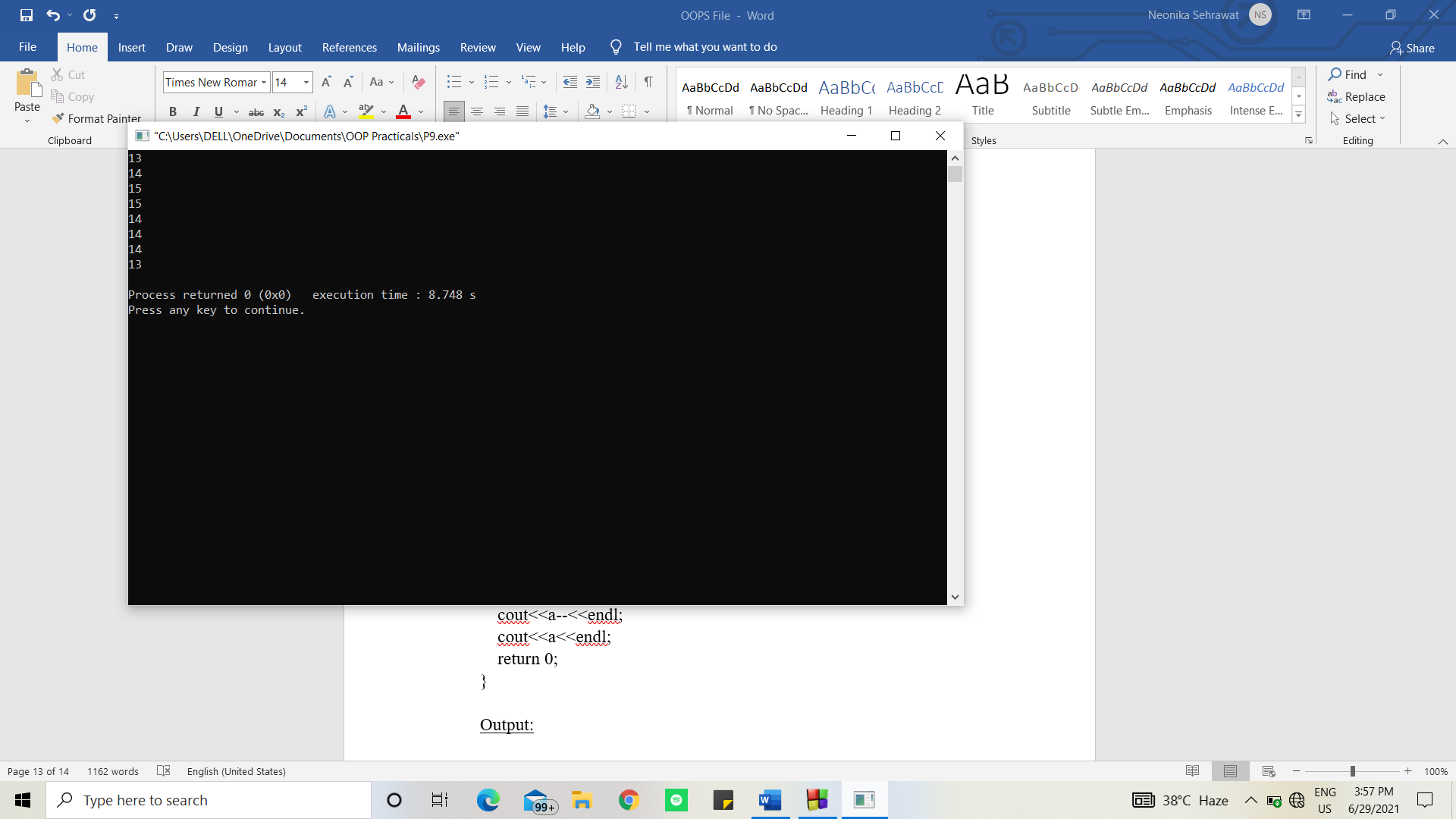
cout<<a--<<endl;

cout<<a<<endl;

return 0;

}

Output:



1. *Write a program to demonstrate the multilevel inheritance.*

#include <iostream>

using namespace std;

class base //single base class

{

public:

int x;

void getdata()

{

cout << "Enter value of x= "; cin >> x;

}

};

class derive1 : public base // derived class from base class

{

public:

int y;

void readdata()

{

cout << "\nEnter value of y= "; cin >> y;

}

};

class derive2 : public derive1 // derived from class derive1

{

private:

int z;

public:

void indata()

{

cout << "\nEnter value of z= "; cin >> z;

}

void product()

{

cout << "\nProduct= " << x \* y \* z;

}

};

int main()

{

derive2 a; //object of derived class

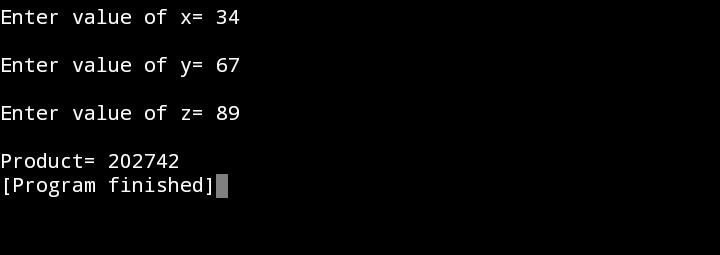
a.getdata();

a.readdata();

a.indata();

a.product();

return 0;

}

1. *Write a program to demonstrate the multiple inheritance*

#include<iostream>

using namespace std;

class A

{

public:

A() { cout << "Neonika's first constructor called" << endl; }

};

class B

{

public:

B() { cout << "Neonika's second constructor called" << endl; }

};

class C: public B, public A // Note the order

{

public:

C() { cout << "Neonika's third constructor called" << endl; }

};

int main()

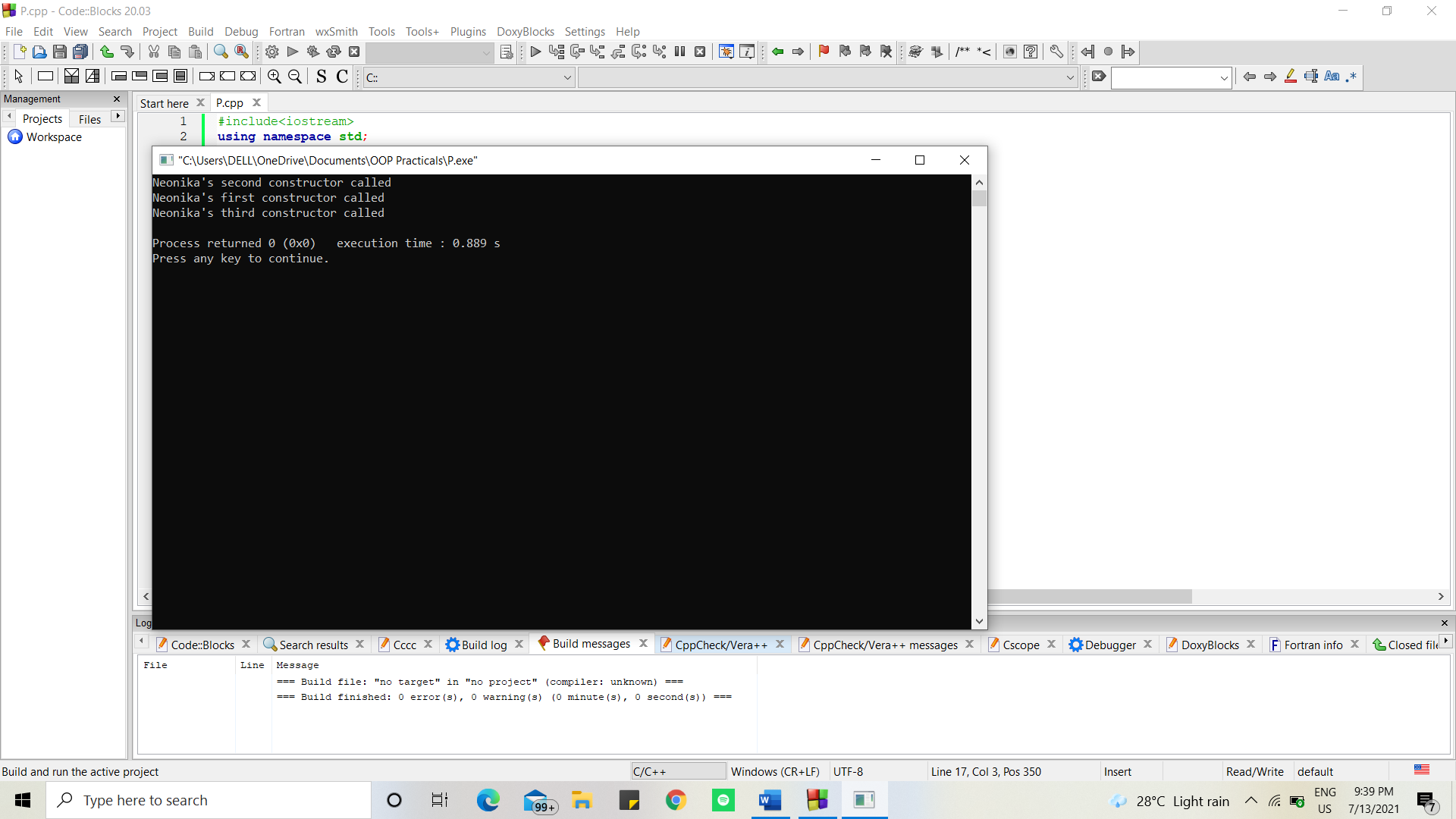
{

C c;

return 0;

}

Output:



1. *Write a program to demonstrate the virtual derivation of a class.*

#include <iostream>

using namespace std;

class A {

public:

int a;

A() // constructor

{

a = 100;

}

};

class B : public virtual A {

};

class C : public virtual A {

};

class D : public B, public C {

};

int main()

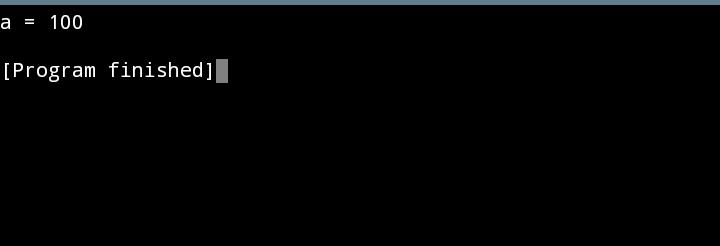
{

D object; // object creation of class d

cout << "a = " << object.a << endl;

return 0;

}

Output:

1. *Write a program to demonstrate the runtime polymorphism.*

#include <iostream>

using namespace std;

class Base {

public:

virtual void show()

{

cout << " In Base \n";

}

};

class Derived : public Base {

public:

void show()

{

cout << "In Derived \n";

}

};

int main(void)

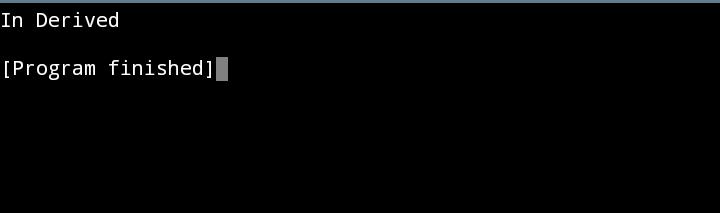
Base\* bp = new Derived;

// RUN-TIME POLYMORPHDerive

bp->show();

return 0;

}

Output:

1. *Write a program to demonstrate the exception handling.*

#include <iostream>

#include <conio.h>

using namespace std;

int main()

{

int a,b;

cout << "Enter 2 numbers: ";

cin >> a >> b;

try

{

if (b != 0)

{

float div = (float)a/b;

if (div < 0)

throw 'e';

cout << "a/b = " << div;

}

else

throw b;

}

catch (int e)

{

cout << "Exception: Division by zero";

}

catch (char st)

{

cout << "Exception: Division is less than 1";

}

catch(...)

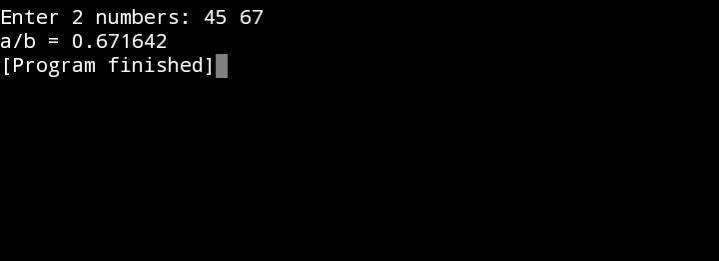
{

cout << "Exception: Unknown";

}

getch();

return 0;}

Output:

1. *Write a program to demonstrate the use of function template.*

#include<iostream>

using namespace std;

float funcAverage(int a, int b){

float avg= (a+b)/2.0;

return avg;

}

int main(){

float a;

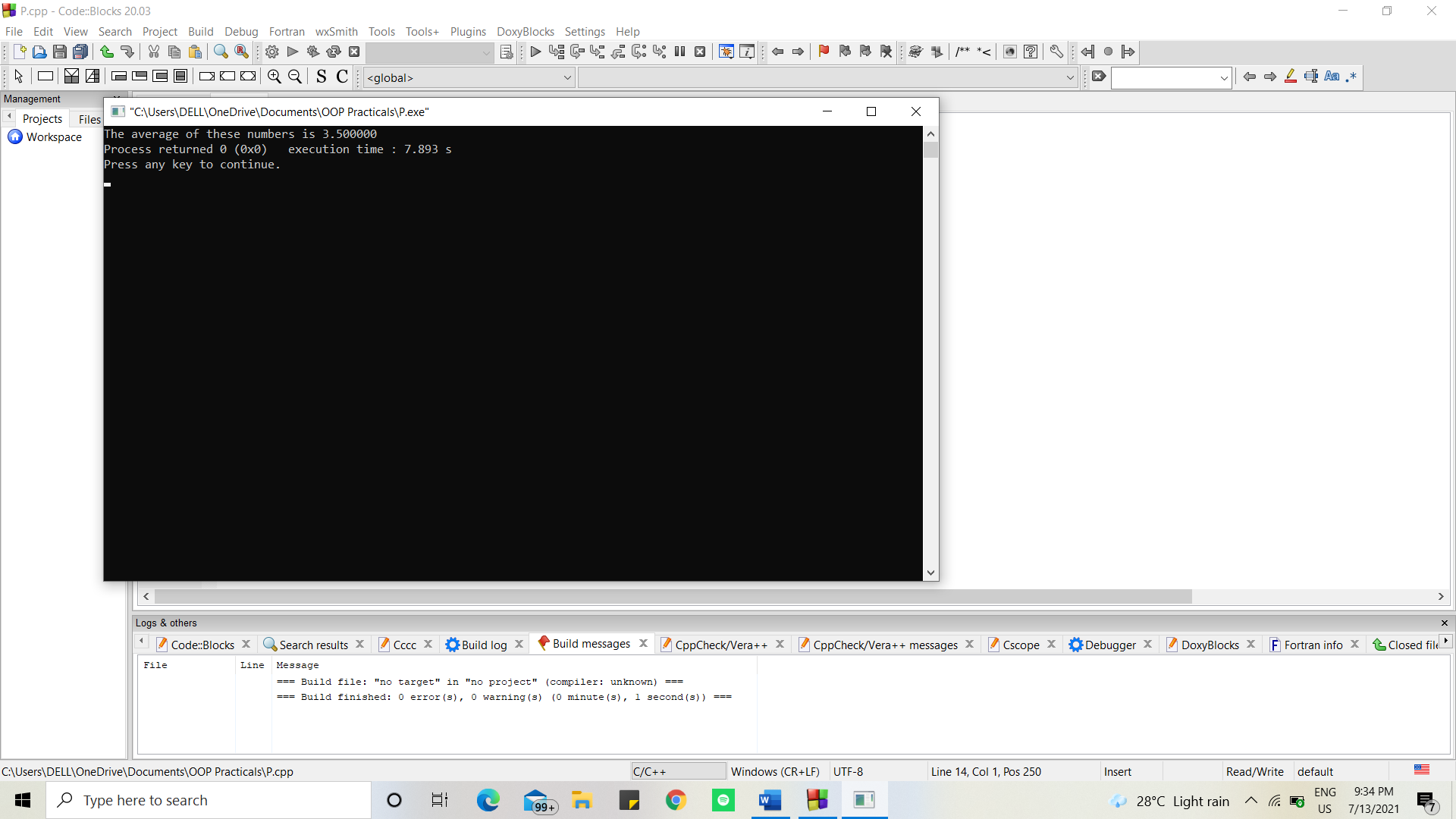
a = funcAverage(5,2);

printf("The average of these numbers is %f",a);

return 0;

}

Output:



1. *Write a program to demonstrate the use of class template.*

#include <iostream>

using namespace std;

class vector

{

public:

int \*arr;

int size;

vector(int m)

{

size = m;

arr = new int[size];

}

int dotProduct(vector &v){

int d=0;

for (int i = 0; i < size; i++)

{

d+=this->arr[i]\*v.arr[i];

}

return d;

}

};

int main()

{

vector v1(3); //vector 1

v1.arr[0] = 4;

v1.arr[1] = 3;

v1.arr[2] = 1;

vector v2(3); //vector 2

v2.arr[0]=1;

v2.arr[1]=0;

v2.arr[2]=1;

int a = v1.dotProduct(v2);

cout<<a<<endl;

return 0;

}

Output:

