**Software Development Lab – II [15B17CI271]**

**Assignment Sheet**

**Week 5**

Q1) Write a C++ program given that there are two base classes namely class A and class B from which class C is inherited. The class A contains member function getBase() and reads “Base” value as user input from keyboard. Class B contains member function getHeight() and reads “Height” value as user input from keyboard. The derived class C inherits all the public members of A and B and computes the area of the triangle.

#include<iostream>

using namespace std;

class A

{

public :

float base ;

void get\_base()

{

cout<<"Enter the base length of the triangle\n";

cin>>base;

}

};

class B

{

public :

float height;

void get\_height()

{

cout<<"Enter the height of the triangle\n";

cin>>height;

}

};

class C : public A , public B

{

public:

float area;

void put\_area()

{

area=0.5\*(base\*height);

cout<<"The area of the triangle is : "<<area<<" squnits\n";

}};

int main()

{

C c1;

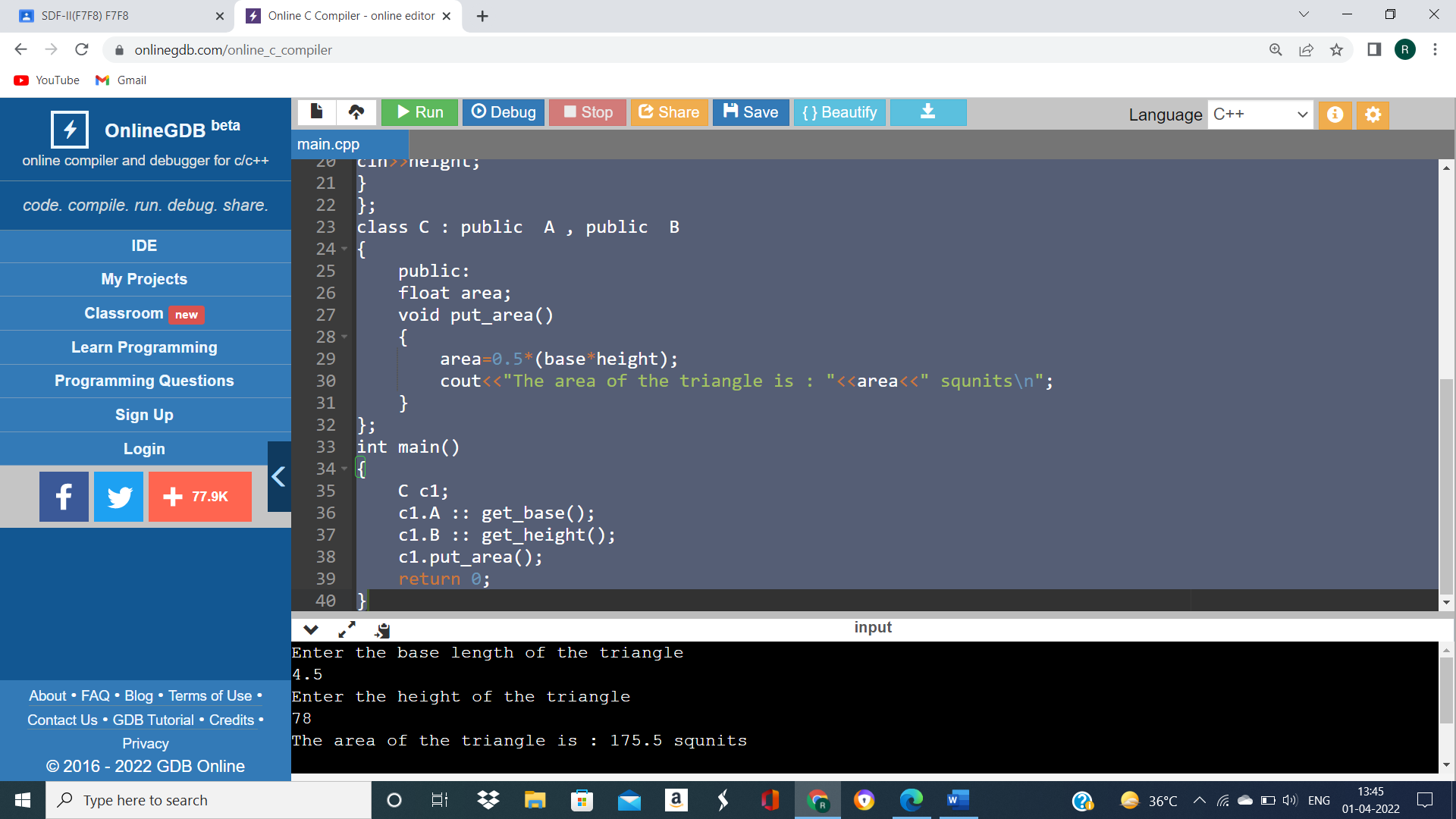
c1.A :: get\_base();

c1.B :: get\_height();

c1.put\_area();

return 0;

}



Q2) Write a C++ program, consider that there are two base classes namely class StudentsDetails and class Marks from which class C is inherited. The class A contains member function getDetails() that reads “students name”, “Enrollment number” value as user input from keyboard. Class B contains member function getMarks() and reads “5 subject marks” value as user input from keyboard.

#include<iostream>

using namespace std;

class student\_details

{

public :

string name;

int enroll;

void get\_details()

{

cout<<"Enter the name of the student \n";

getline(cin,name);

cout<<"Enter the enrollment no.\n";

cin>>enroll;

}

};

class marks

{

public:

int j;

int arr[5];

void get\_marks()

{

cout<<"enter marks";

for( j = 0; j<5;j++)

{

cin>>arr[j];

}

}

};

class C : public student\_details , public marks

{

public :

int k;

int total=0;

void display()

{

for( k = 0;k<5;k++)

total=total+arr[k];

cout<<"Name : "<<name<<endl;

cout<<"Enrollment no. : "<<enroll<<endl;

cout<<"Total marks : "<<total<<endl;

}

};

int main()

{

C c1;

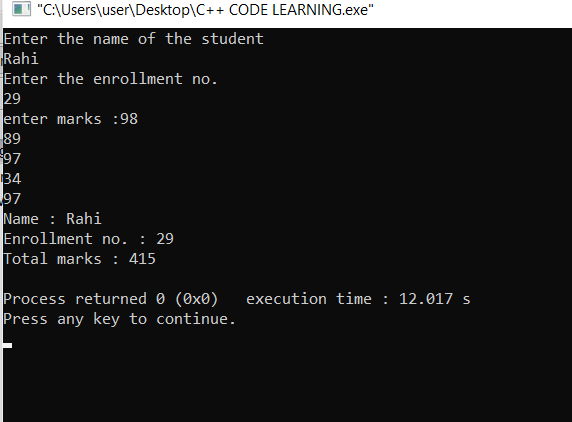
c1.student\_details::get\_details();

c1.marks::get\_marks();

c1.display();

return 0;

}



Q3) Based on the virtual function concept, write the main function for the following code to display the derived class values given by user at run time.

#include <iostream>

using namespace std;

class base

{

public:

char fname[20];

char surname[20];

public:

virtual void calculate()

{

cout << "enter fname:";

cin>> fname;

cout << "enter surname";

cin >> surname;

}

void display()

{

cout << "welcome " << fname <<" "<< surname<<endl;

}

};

class derived : public base {

public:

void calculate()

{

cout << "enter derived\_fname:";

cin>>fname;

cout << "enter derived\_ surname";

cin>>surname;

}

void display()

{

cout << "welcome to derived " << fname <<" "<< surname<<endl;

}

};

int main()

{

base b1;

derived d1;

b1.calculate();

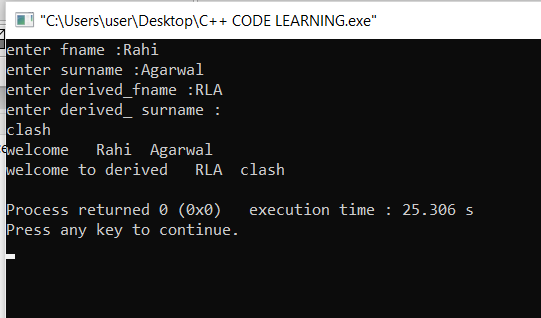
d1.calculate();

b1.display();

d1.display();

return 0;

}



Q4) Given a snippet of the program to create a base class named as base\_food\_Items with a virtual function named as order and total\_Price . Create a derived class name Chinese. Then calculate the total\_price of food items based on variables quantity and item\_price.

#include<iostream>

using namespace std;

class base\_food\_items

{

public:

char item\_name[20];

int quantity ;

int price;

public :

virtual void order()

{

cout<<"Enter item name\n";

cin>>item\_name;

cout<<"Enter quantity\n";

cin>>quantity;

cout<<"Item price\n";

cin>>price;

}

void total\_price()

{

cout<<"Order is :"<<item\_name<<"\t"<<"Quantity: "<<quantity<<endl;

cout<<"Total price = "<<price<<endl;

}

};

class chinese : public base\_food\_items

{

};

int main()

{

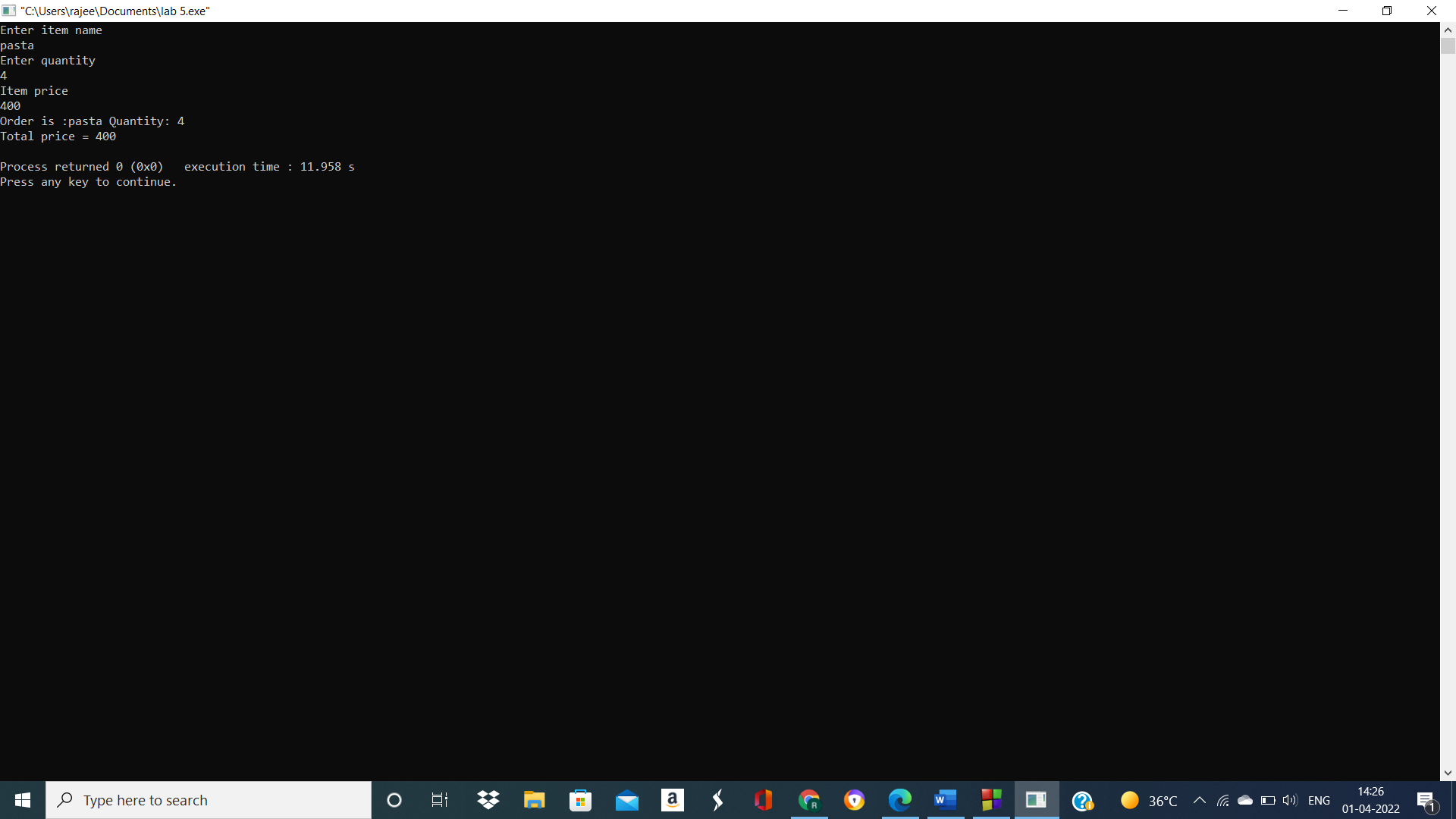
chinese c1;

c1.order();

c1.total\_price();

return 0;

}



Q5) Write a C++ program to show the functionality of the abstract classes.

#include <iostream>

using namespace std;

class A

{

public:

virtual void display1() = 0;

virtual void display2() = 0;//declaring a virtual function

};

class B : public A

{

public:

void display1()

{

cout << "This is Display1() method of Derived Class" << endl;

}

void display2()

{

cout << "This is Display2() method of Derived Class" << endl;

}

};

int main(void)

{

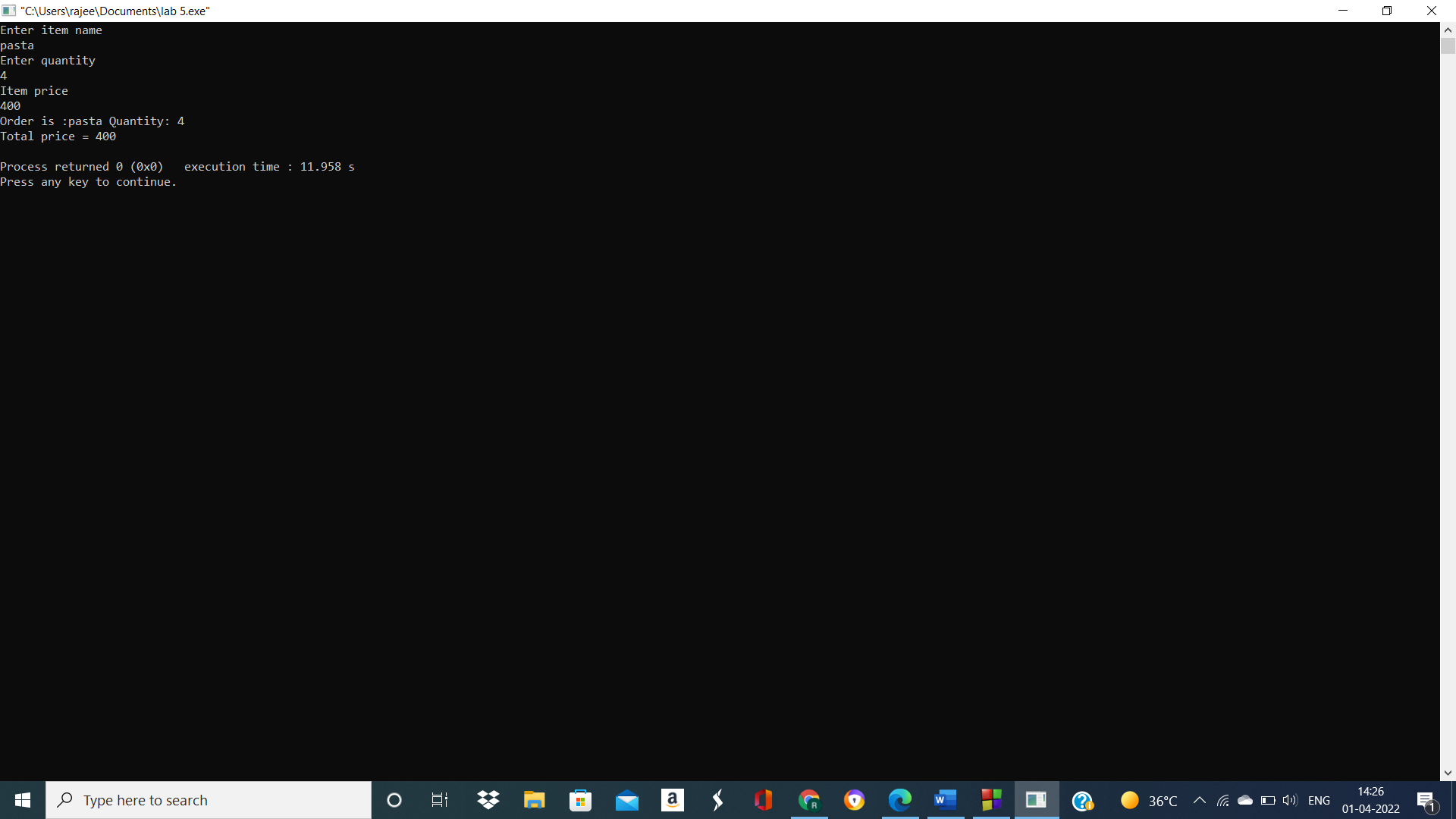
B obj;

obj.display1();

obj.display2();

return 0;

}



Q6) Write a program to use constructors of the abstract class to find the sum of two numbers and display the results.

#include <iostream>

using namespace std;

class A

{

public:

int a, b, c;

virtual void test() = 0;

A(int a, int b)

{

cout << "Hello I am the constructor" << endl;

}};

class B : public A

{

int y;

public:

B(int i, int j) : A(a, b)

{

a = i;

b = j;

c = a + b;

}

void test() { cout << "The sum is = " << c << endl; }

};

int main(void)

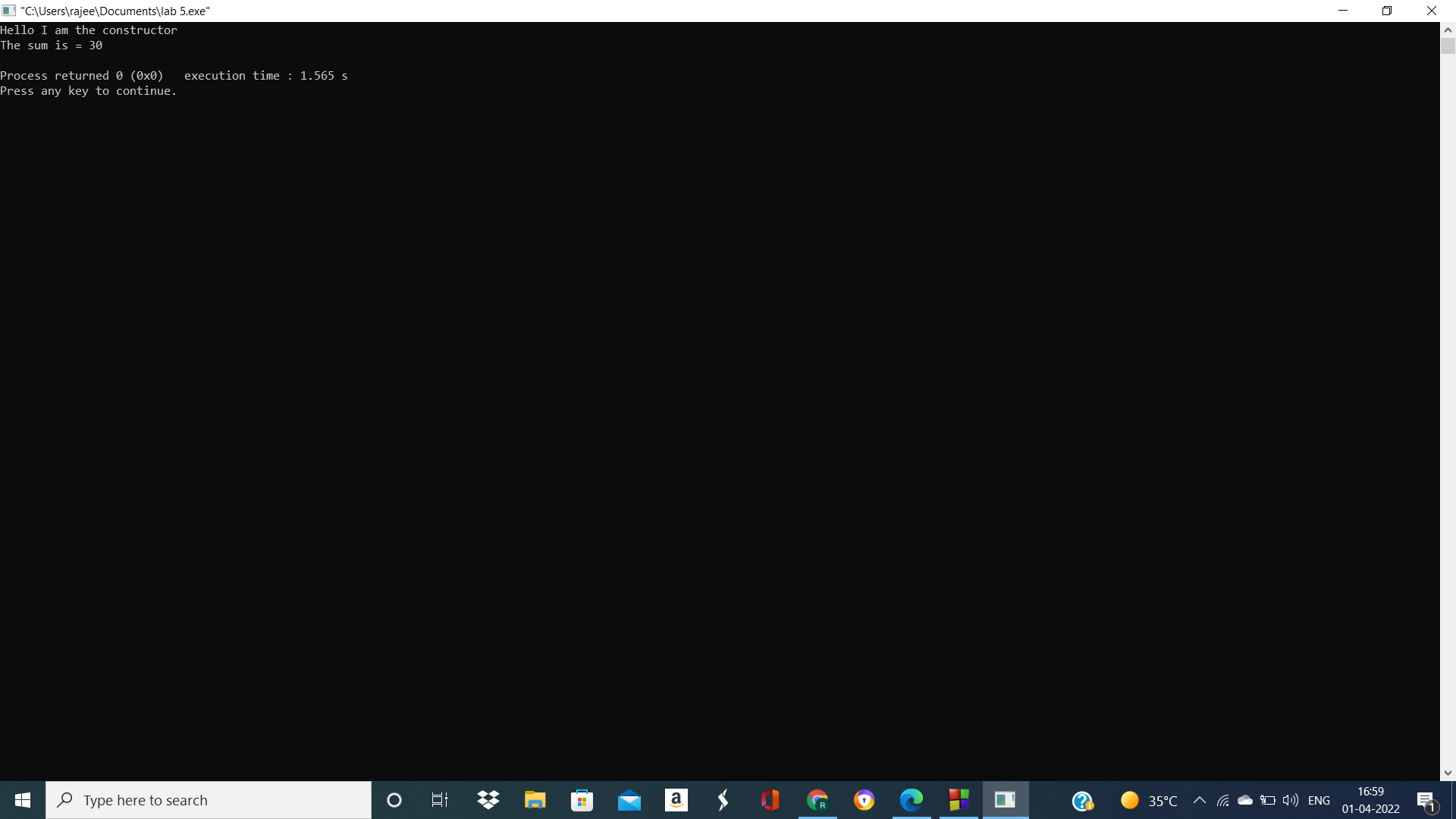
{

B obj(10, 20);

obj.test();

return 0;

}



Q7 A) What is the output of the following code?

#include<iostream>

using namespace std;

class Base { };

class Derived: public Base {};

int main() {

Base \*base\_ptr = new Derived;

Derived \*derived\_ptr = dynamic\_cast<Derived\*>(base\_ptr);

if(derived\_ptr != NULL)

cout<<"It is working";

else

cout<<"cannot cast Base\* to Derived\*";

return 0;

}

OUTPUT:-

error: cannot dynamic\_cast 'base\_ptr' (of type 'class Base\*') to type 'class Derived\*' (source type is not polymorphic)|

Q7 B) Rewrite the above code to rectify the error

#include<iostream>

using namespace std;

class Base

{

virtual void function()

{}};

class Derived: public Base {};

int main()

{

Base \*base\_ptr = new Derived;

Derived \*derived\_ptr = dynamic\_cast<Derived\*>(base\_ptr);

if(derived\_ptr != NULL)

cout<<"It is working";

else

cout<<"cannot cast Base\* to Derived\*";

return 0;

}

