**Program 12/Task-01**

**Source Code:**

#include<iostream>

using namespace std;

class parent{

    public:

    parent(){

        cout<<"Parent Constructor is called."<<endl;

    }

};

class child: public parent{

    public:

    child(){

        cout<<"Child Constructor is called."<<endl;

    }

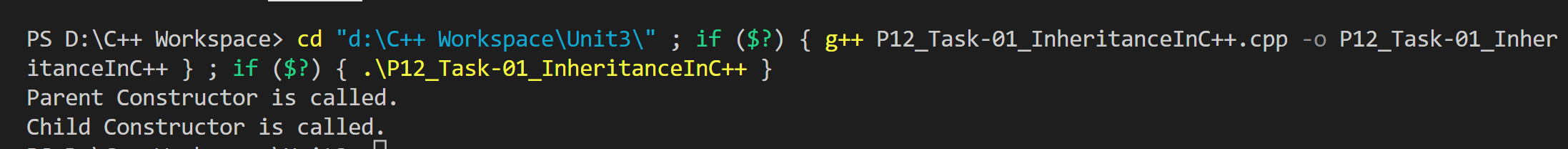
};

int main(){

    child x;

}

**Output**



**Program 12/Task-03**

**Source Code:**

#include<iostream>

using namespace std;

class Mammals{

    public:

    void printm1(){

        cout<<"I am mammal"<<endl;

    }

};

class MarineAnimals{

    public:

    void printm2(){

        cout<<"I am the marine animal"<<endl;

    }

};

class BlueWhale: public Mammals,public MarineAnimals{

        public:

        void printm3(){

            cout<<"I belong to both the categories:Mammals as well as Marinie Animals"<<endl;

        }

};

int main(){

    Mammals x;

    MarineAnimals y;

    BlueWhale z;

    x.printm1();

    y.printm2();

    z.printm3();

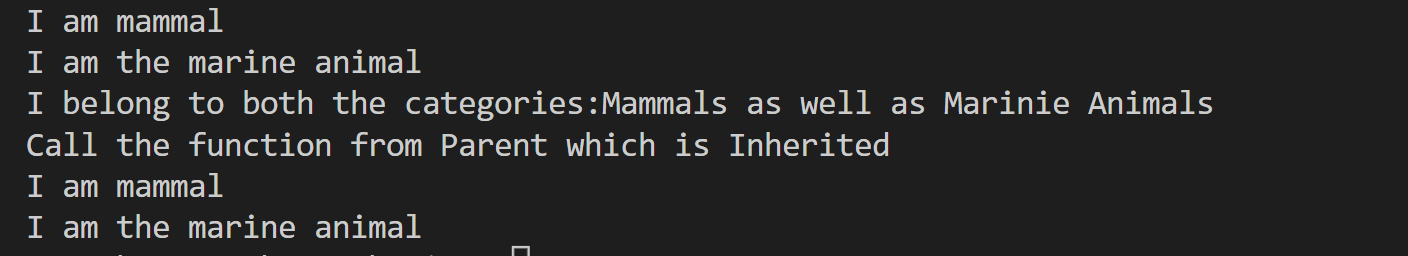
    cout<<"Call the function from Parent which is Inherited"<<endl;

    z.printm1();

    z.printm2();

}

**Output**



**Program 13/Task-01**

**Source Code:**

#include<iostream>

using namespace std;

class A{

    public:

    void display(){

        cout<<"Parent class\n";

    }

};

class B{

    public:

      void display(){

        cout<<"Parent class\n";

    }

};

class C: public A,public B{

    public:

};

int main(){

    C obj;

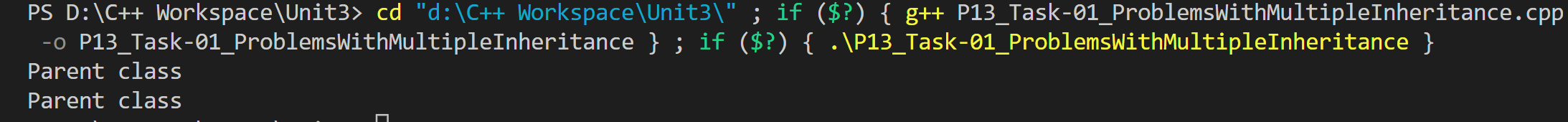
   // obj.display(); =>  error: request for member 'display' is ambiguous obj.display();

        obj.A::display();

        obj.B::display();

}

**Output**



**Program 13/Task-03**

**Source Code:**

#include<iostream>

using namespace std;

class base{

    public:

    int x;

    base(){

        this->x=10;

    }

};

class derived1:public base{

};

class derived2:protected base{

};

class derived3:private base{

};

int main(){

    derived1 obj1;

    cout<<obj1.x<<endl;

   /\*

   derived2 obj2;

    cout<<obj2.x<<endl; 'int base::x' is inaccessible within this context

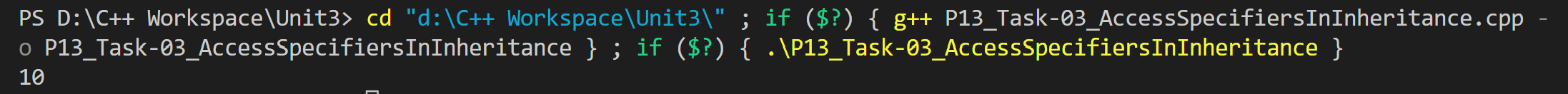
    derived3 obj3;

    cout<<obj3.x<<endl; error: 'int base::x' is inaccessible within this context

    \*/

}

**Output**



**Program 14/Task-01**

**Source Code:**

#include<iostream>

using namespace std;

class base{

        public:

        void Add(int a,int b){

           cout<<"Sum of addition is:"<<(a+b)<<endl;

        }

        void Add(int a,float b){

                cout<<"Sum of addition is:"<<(a+b)<<endl;

        }

};

class child: public base{

        public:

        void Add(int a,float b){

                cout<<"Sum of addition is:"<<(a+b+10)<<endl;

        }

};

int main(){

        base obj;

        child obj2;

        obj.Add(10,20);

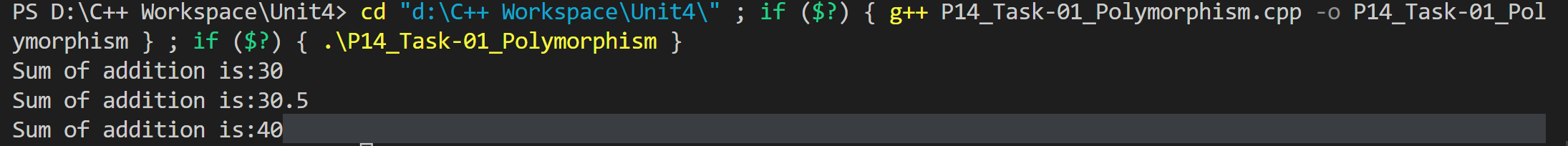
        //obj.Add(10,20.5);       //It is ambiguous

        obj.Add(10,20.5f);       //Overloaded function in base class

        obj2.Add(10,20);        //Overridden function is called

}

**Output**



**Program 14/Task-02**

**Source Code:**

#include<iostream>

using namespace std;

class base

{

    public:

    virtual double add(int a,double b)

    {

        return a+b;

    }

};

class child:public base{

    public:

    double add(int a,double b)

    {

        return a+b+7.9;

    }

};

int main()

{

  base \*p;

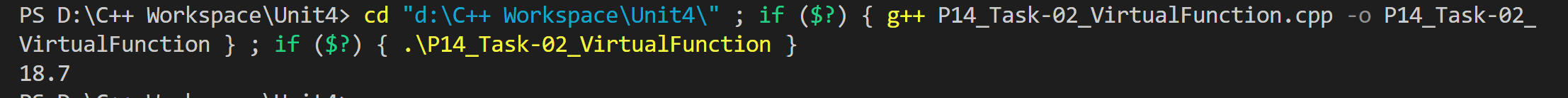
  child d;

  p=&d;

  cout<<p->add(3,7.8)<<endl;

}

**Output**



**Program 15/Task-01**

**Source Code:**

#include<iostream>

using namespace std;

class base{

public:

virtual int add(int a,int b){}

};

class child:public base{

public:

int add(int a,int b){

return (a+b);

}

};

int main(){

base \*ob=new child();

// float b;

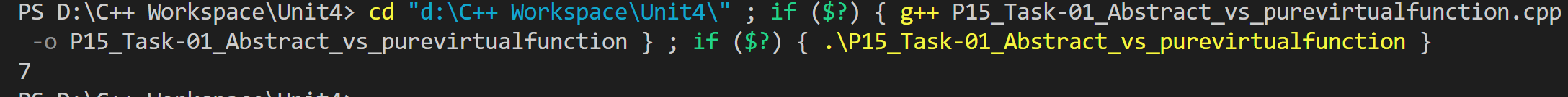
// cin>>b;

cout<< ob->add(2,5);

return 0;

}

**Output**



**Program 16/Task-01**

**Source Code:**

#include<iostream>

using namespace std;

class base

{

    public:

    virtual void display()

    {

        cout<<"Display of base class"<<endl;

    }

     void display(int b)

     {

        cout<<"Display of base class:"<<b<<endl;

     }

};

class child:public base

{

    public:

    void display()

    {

        cout<<"Display of chlid class"<<endl;

    }

};

int main()

{

    base \*p;

    child x;

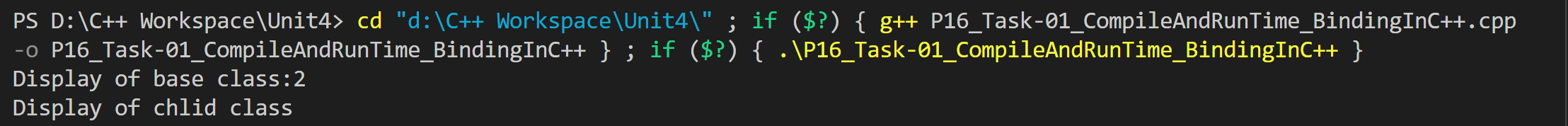
    p=&x;

    p->display(2);

    p->display();}

}

**Output**



**Program 17/Task-01**

**Source Code:**

#include<iostream>

#include<list>

using namespace std;

int main(){

    list<int> l={5,2,10,4,5};

    for(auto it=begin(l);it!=end(l);it++){

        cout<<\*it<<endl;

    }

    cout<<"Size of list is:"<<l.size()<<endl;

    l.sort();

    cout<<"sorted list:"<<endl;

     for(auto it=begin(l);it!=end(l);it++){

        cout<<\*it<<endl;

    }

    l.reverse();

    cout<<"Reversed list:"<<endl;

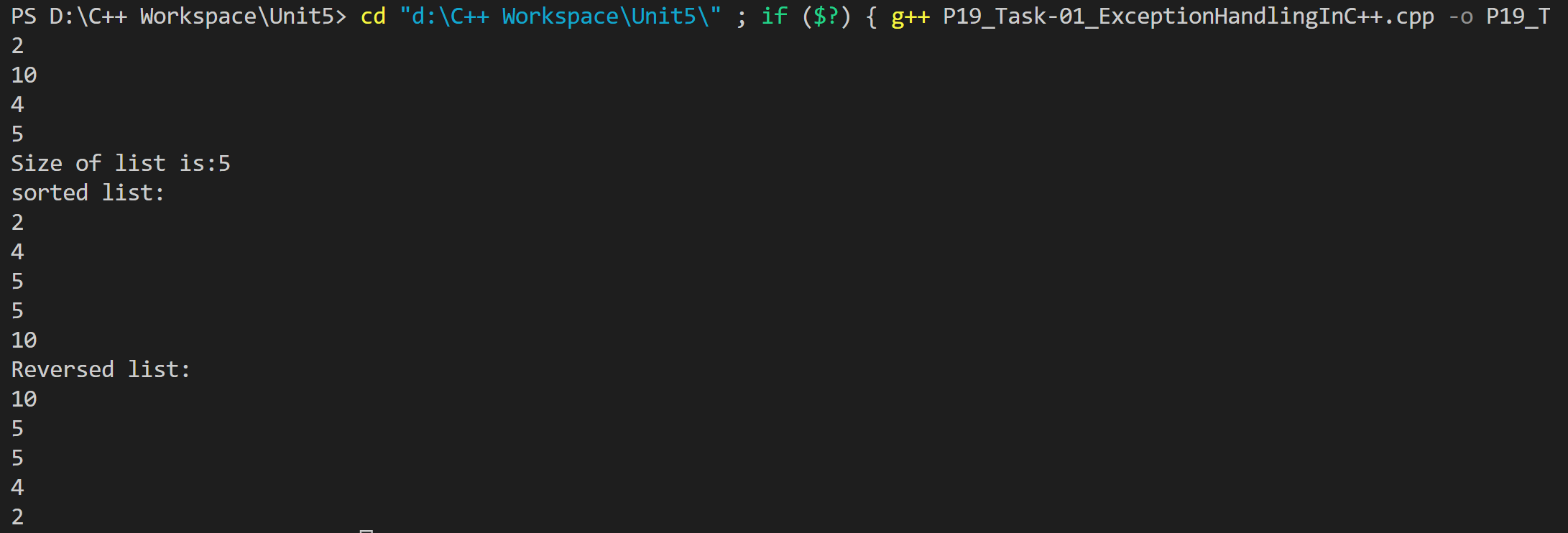
    for(auto it=begin(l);it!=end(l);it++){

        cout<<\*it<<endl;

    }

}

**Output**



**Program 17/Task-02**

**Source Code:**

#include<iostream>

#include<vector>

using namespace std;

int main(){

    /\*vector<int> v={1,2,3,4,5};

    for(auto it=v.begin();it<v.end();it++){

        cout<<\*it<<" ";

    }

    cout<<endl;

    cout<<"Size of vector:"<<v.size();

    cout<<"Capacity of v"

    \*/

   vector<int> g1;

   for(int i=1;i<=5;i++){

    g1.push\_back(i);;

   }

   cout<<"Output of begin and end:";

   for(auto i=g1.begin();i!=g1.end();++i){

    cout<<\*i<<" ";

   }

   cout<<"\nOutput of rbegin and rend:";

   for(auto ir=g1.rbegin();ir!=g1.rend();++ir){

    cout<<\*ir<<" ";

   }

   cout<<endl;

   cout<<"Size of vector:"<<g1.size()<<endl;

    cout<<"Capacity of vector:"<<g1.capacity()<<endl;

    cout<<"Max\_size:"<<g1.max\_size()<<endl;

    g1.resize(4);

    cout<<"Size:"<<g1.size();

    if(g1.empty()==false)

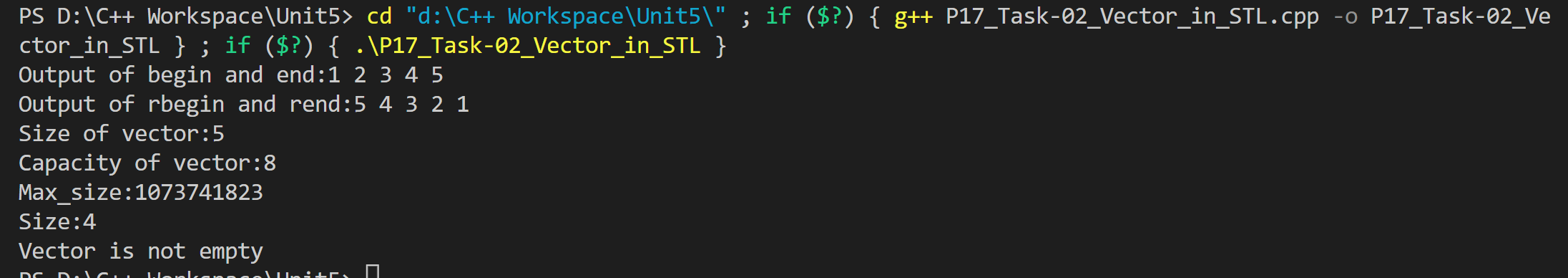
        cout<<"\nVector is not empty"<<endl;

    else

        cout<<"Vector is empty";

}

**Output**



**Program 18/Task-01**

**Source Code:**

#include<iostream>

#include <fstream>

using namespace std;

int main(){

    ofstream file;

    string data;

    file.open("example.txt");

    file<<"Hello,How are you?";

    file.close();

    ifstream file2;

    file2.open("example.txt");

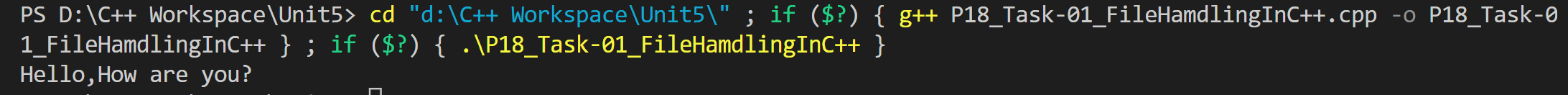
    getline(file2,data);

    cout<<data;

    file2.close();

}

**Output**



**Program 19/Task-01**

**Source Code:**

#include<iostream>

using namespace std;

double division(int a,int b){

    try{

        if(b==0){

            throw "Division by zero condition!";

        }

    }

    catch(const char\* msg){

        throw;

    }

    return (a/b);

}

int main(){

    int x=50;

    int y=0;

    double z=0;

    try{

        z=division(x,y);

        cout<<z<<endl;

    }catch(const char\* msg){

        cerr<<msg<<endl;

    }

    //2

    try{

        throw 'a';

    }

    catch(int x){

        cout<<"Caught"<<x;

    }

    catch(...){

        cout<<"Default Exception\n";

    }

    //3

    try{

        throw 'a';

    }

    catch(int x){

        cout<<"Caught";

    }

    return 0;

}

**Output**

