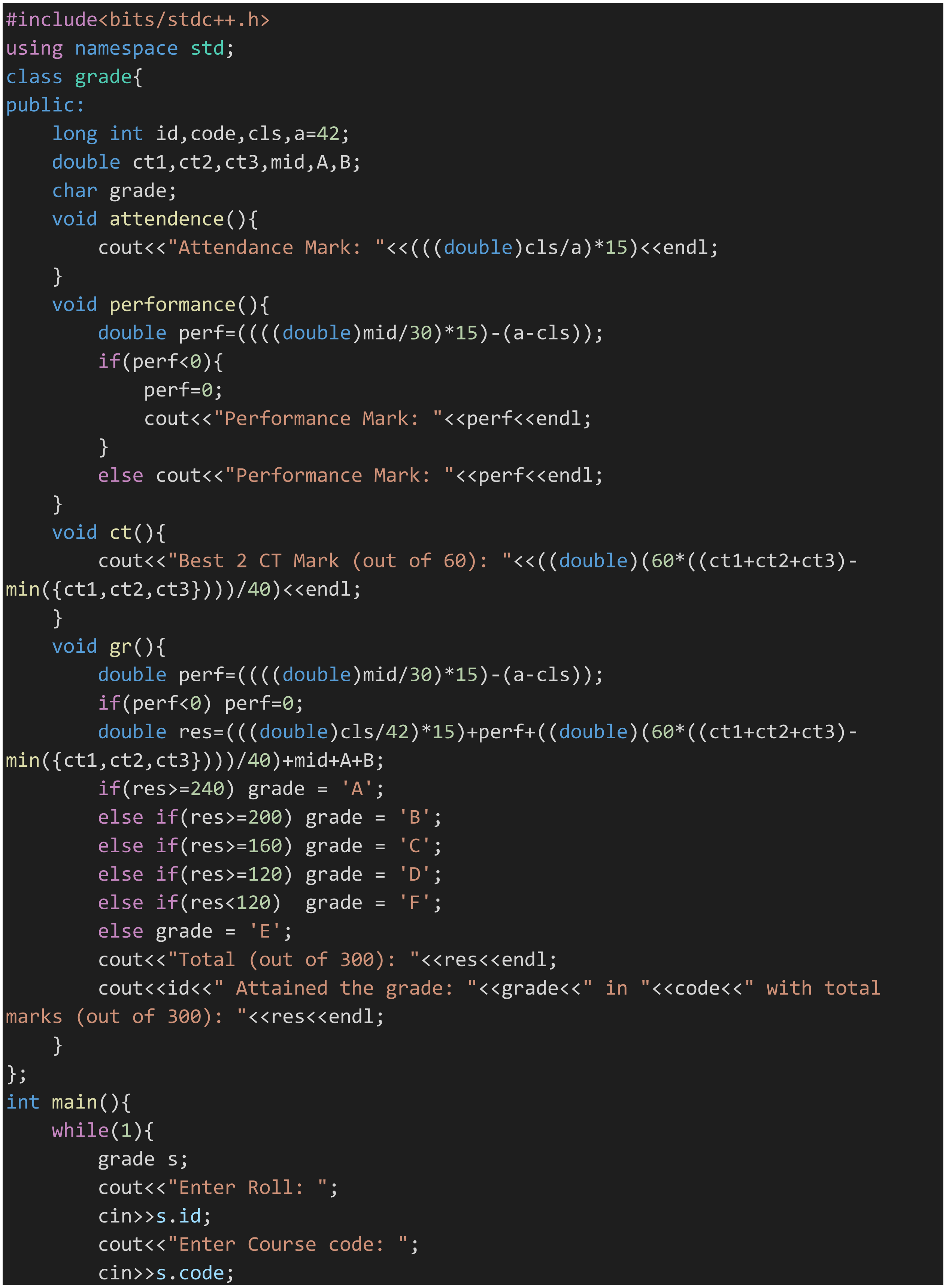
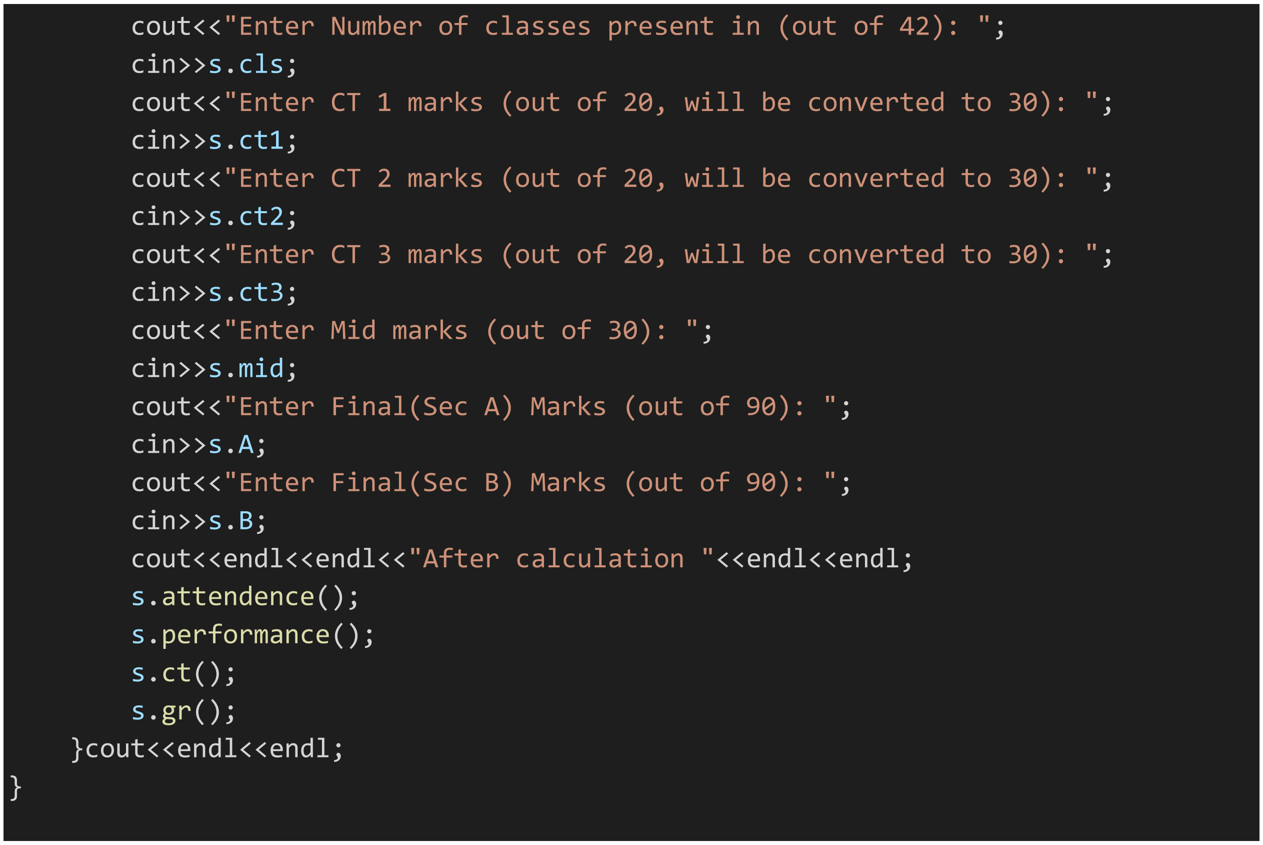


```
#include<bits/stdc++.h>
using namespace std;
class grade{
public:
    long int id,code,cls,a=42;
    double ct1,ct2,ct3,mid,A,B;
    char grade;
    void attendence(){
        cout<<"Attendance Mark: "<<(((double)cls/a)*15)<<endl;
    }
    void performance(){
        double perf=((((double)mid/30)*15)-(a-cls));
        if(perf<0){
            perf=0;
            cout<<"Performance Mark: "<<perf<<endl;
        }
        else cout<<"Performance Mark: "<<perf<<endl;
    }
    void ct(){
        cout<<"Best 2 CT Mark (out of 60): "<<((double)(60*((ct1+ct2+ct3)-
min({ct1,ct2,ct3})))/40)<<endl;
    }
    void gr(){
        double perf=((((double)mid/30)*15)-(a-cls));
        if(perf<0) perf=0;
        double res=(((double)cls/42)*15)+perf+((double)(60*((ct1+ct2+ct3)-
min({ct1,ct2,ct3})))/40)+mid+A+B;
        if(res>=240) grade = 'A';
        else if(res>=200) grade = 'B';
        else if(res>=160) grade = 'C';
        else if(res>=120) grade = 'D';
        else if(res<120)  grade = 'F';
        else grade = 'E';
        cout<<"Total (out of 300): "<<res<<endl;
        cout<<id<<" Attained the grade: "<<grade<<" in "<<code<<" with total
marks (out of 300): "<<res<<endl;
    }
};
int main(){
    while(1){
        grade s;
        cout<<"Enter Roll: ";
        cin>>s.id;
        cout<<"Enter Course code: ";
        cin>>s.code;
```



```
        cout<<"Enter Number of classes present in (out of 42): ";
        cin>>s.cls;
        cout<<"Enter CT 1 marks (out of 20, will be converted to 30): ";
        cin>>s.ct1;
        cout<<"Enter CT 2 marks (out of 20, will be converted to 30): ";
        cin>>s.ct2;
        cout<<"Enter CT 3 marks (out of 20, will be converted to 30): ";
        cin>>s.ct3;
        cout<<"Enter Mid marks (out of 30): ";
        cin>>s.mid;
        cout<<"Enter Final(Sec A) Marks (out of 90): ";
        cin>>s.A;
        cout<<"Enter Final(Sec B) Marks (out of 90): ";
        cin>>s.B;
        cout<<endl<<endl<<"After calculation "<<endl<<endl;
        s.attendence();
        s.performance();
        s.ct();
        s.gr();
    }cout<<endl<<endl;
}
```

Problem: A line can be constructed by taking multiple line segments, while each line segment consists of 2 coordinates. In this problem, you are given the x and y coordinates of two lines. Your job is to construct the lines and to find the line with the largest length.

Design a class named Line, having one public method to calculate the length of the line.

The skeleton of the class is as follows:

class Line {

// no of points constructing that line (integer value)

// x and y coordinates of those points (real values)

public:

// necessary constructors

calculate_length();

};

Sample Input

Number of points in Line 1: 3

Enter coordinates:

1 1

2 2

3 4

Number of points in Line 2: 2

Enter coordinates:

0 0

5.2 5.6

Sample Output

Length of Line 1: 3.65028

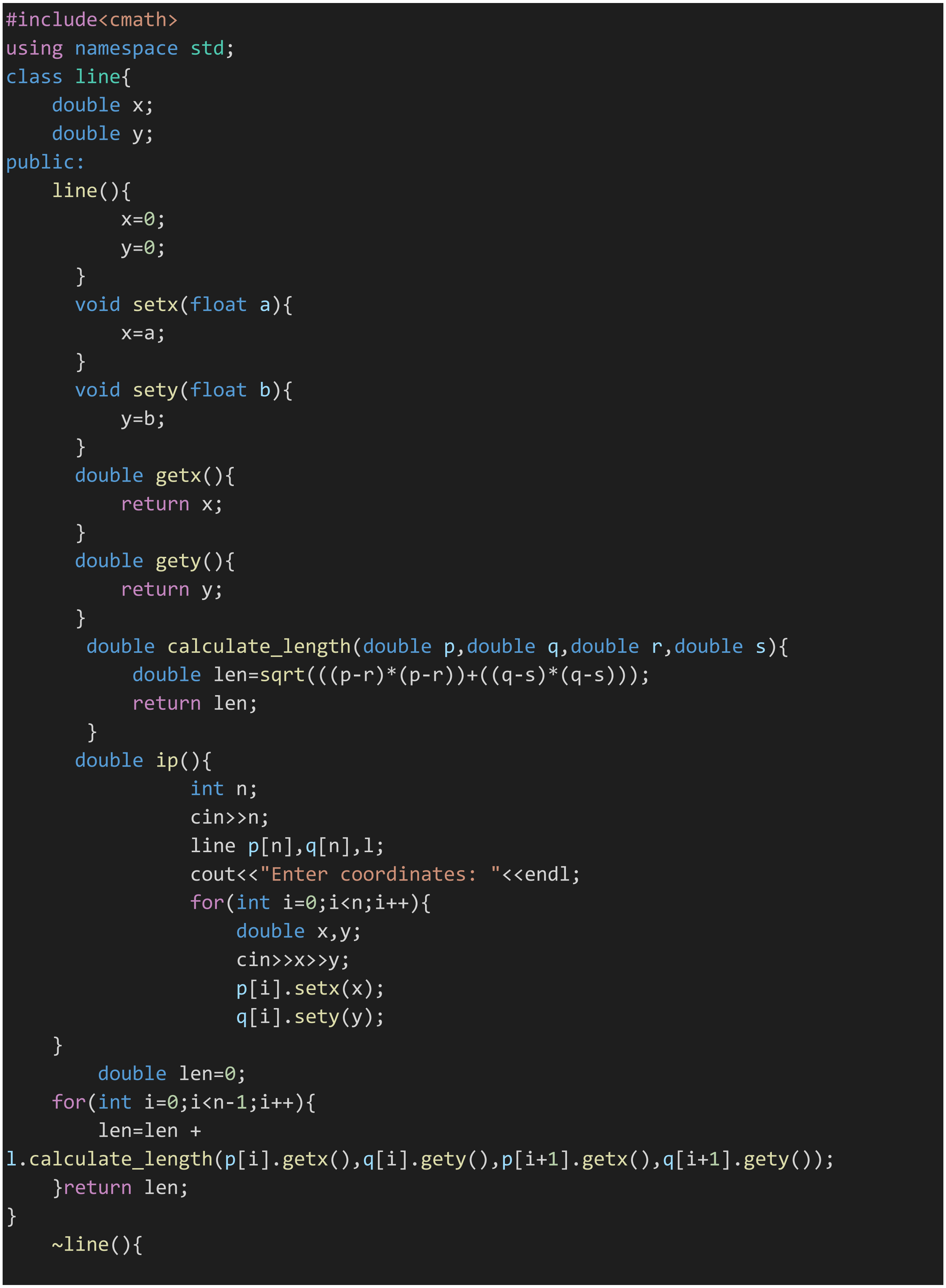
Length of Line 2: 7.64199

Line 2 > Line 1

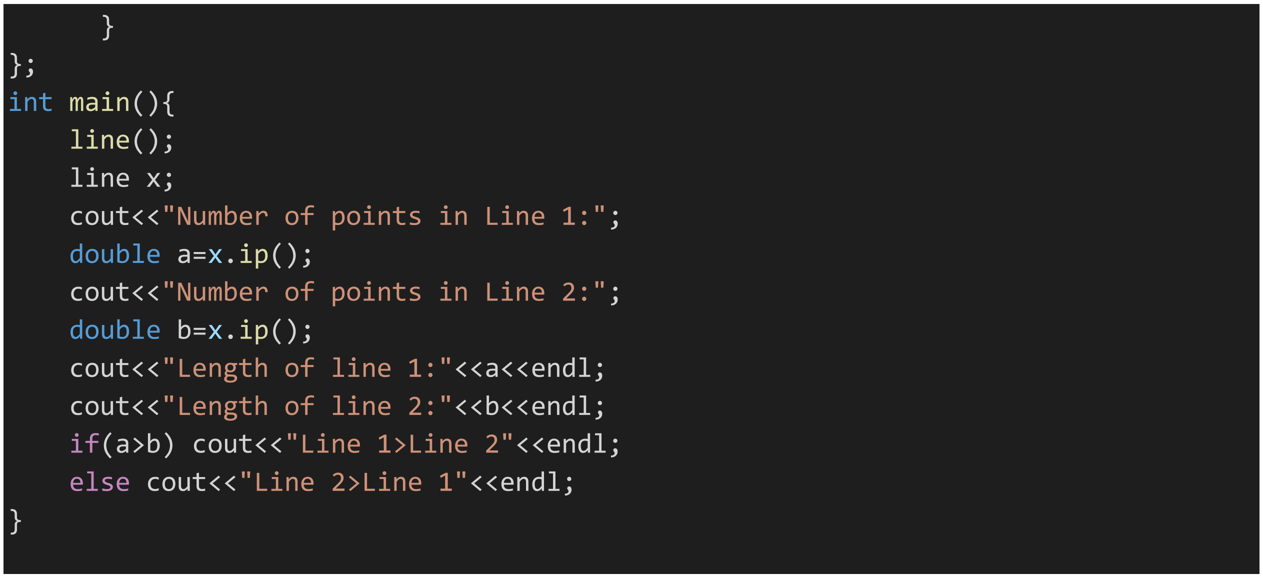
Note: The class should have no additional function (except the constructor) to

set/get/take input of the private variables.

```
/*
Number of points in Line 1: 3
Enter coordinates:
1 1
2 2
3 4
Number of points in Line 2: 2
Enter coordinates:
0 0
5.2 5.6

Length of Line 1: 3.65028
Length of Line 2: 7.64199
Line 2 > Line 1 */
#include<iostream>
```



```
#include<cmath>
using namespace std;
class line{
    double x;
    double y;
public:
    line(){
          x=0;
          y=0;
      }
      void setx(float a){
          x=a;
      }
      void sety(float b){
          y=b;
      }
      double getx(){
          return x;
      }
      double gety(){
          return y;
      }
       double calculate_length(double p,double q,double r,double s){
           double len=sqrt(((p-r)*(p-r))+((q-s)*(q-s)));
           return len;
       }
      double ip(){
                int n;
                cin>>n;
                line p[n],q[n],l;
                cout<<"Enter coordinates: "<<endl;
                for(int i=0;i<n;i++){
                    double x,y;
                    cin>>x>>y;
                    p[i].setx(x);
                    q[i].sety(y);
    }
        double len=0;
    for(int i=0;i<n-1;i++){
        len=len +
l.calculate_length(p[i].getx(),q[i].gety(),p[i+1].getx(),q[i+1].gety());
    }return len;
}
    ~line(){
```



```
      }
};
int main(){
    line();
    line x;
    cout<<"Number of points in Line 1:";
    double a=x.ip();
    cout<<"Number of points in Line 2:";
    double b=x.ip();
    cout<<"Length of line 1:"<<a<<endl;
    cout<<"Length of line 2:"<<b<<endl;
    if(a>b) cout<<"Line 1>Line 2"<<endl;
    else cout<<"Line 2>Line 1"<<endl;
}
```

Write a program that defines a namespace named “Financial”.

The namespace contains another two namespace named “Mortgage” and “Retirement”. The “Mortgage” namespace contains functions related to mortgage payments, including monthlyPayment() and totalPayments() with necessary parameters. The “Retirement” namespace contains functions related to retirement savings, including monthlySavings () and totalSavings() with necessary parameters.

Write a program that takes 'principal_amount', 'interest_rate', and 'number_of_periods' as inputs from the user, and uses these values to calculate the total payments for a mortgage and the total savings needed for retirement.

Calculations of given functions

monthlyPayment ():

r = r / 12.0

c = pow (1.0 + r, n) - 1.0

return (P * r) / c

totalPayments ():

return monthly_payment*n

monthlySavings ():

r= r / 12.0.

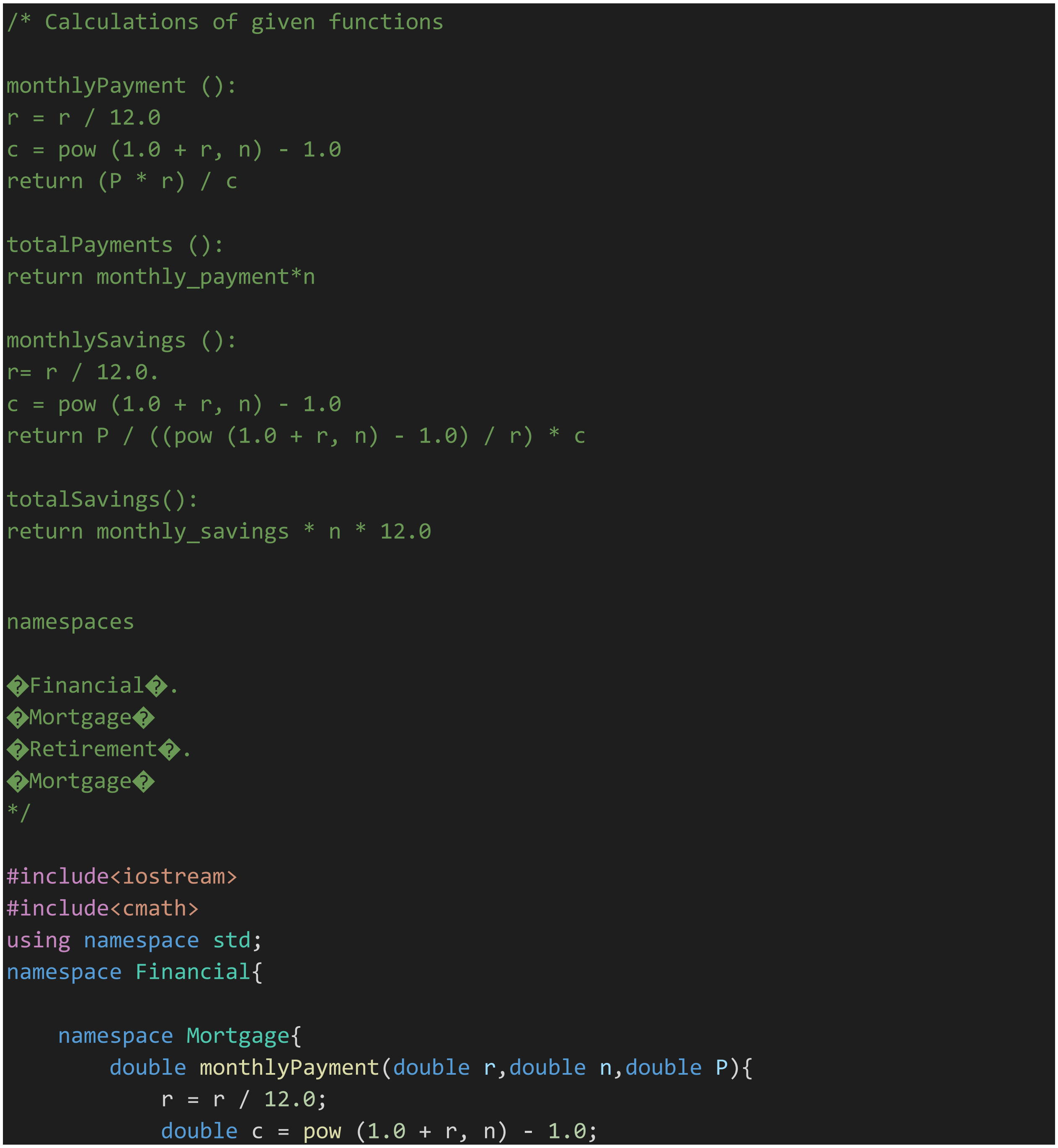
c = pow (1.0 + r, n) - 1.0

return P / ((pow (1.0 + r, n) - 1.0) / r) * c

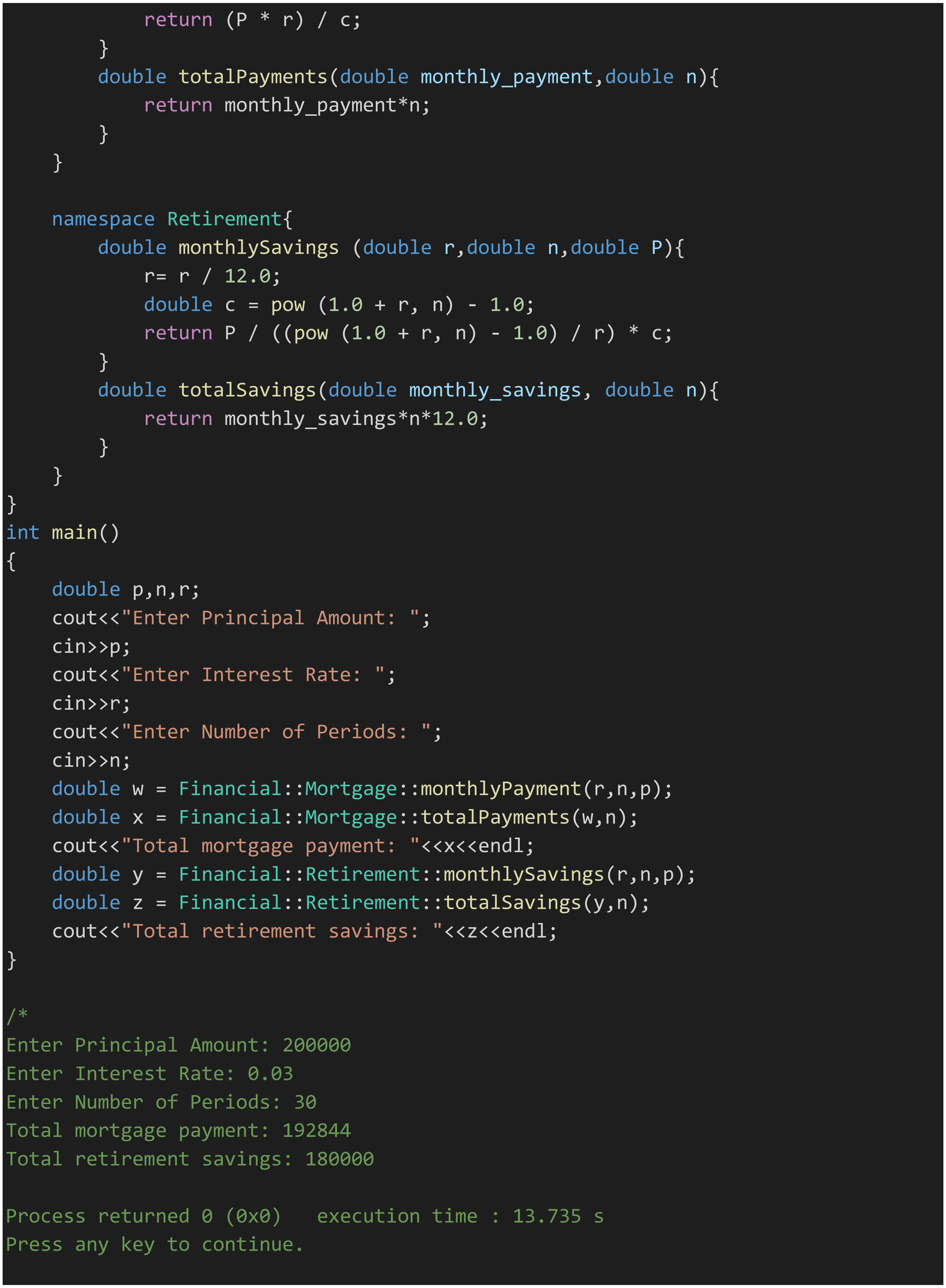
totalSavings():

return monthly_savings * n * 12.0

[NB: P, r, n represent principal_amount, interest_rate, and the number_of_periods respectively]



```
/* Calculations of given functions

monthlyPayment ():
r = r / 12.0
c = pow (1.0 + r, n) - 1.0
return (P * r) / c

totalPayments ():
return monthly_payment*n

monthlySavings ():
r= r / 12.0.
c = pow (1.0 + r, n) - 1.0
return P / ((pow (1.0 + r, n) - 1.0) / r) * c

totalSavings():
return monthly_savings * n * 12.0


namespaces

�Financial�.
�Mortgage�
�Retirement�.
�Mortgage�
*/

#include<iostream>
#include<cmath>
using namespace std;
namespace Financial{

    namespace Mortgage{
        double monthlyPayment(double r,double n,double P){
            r = r / 12.0;
            double c = pow (1.0 + r, n) - 1.0;
```



```
            return (P * r) / c;
        }
        double totalPayments(double monthly_payment,double n){
            return monthly_payment*n;
        }
    }

    namespace Retirement{
        double monthlySavings (double r,double n,double P){
            r= r / 12.0;
            double c = pow (1.0 + r, n) - 1.0;
            return P / ((pow (1.0 + r, n) - 1.0) / r) * c;
        }
        double totalSavings(double monthly_savings, double n){
            return monthly_savings*n*12.0;
        }
    }
}
int main()
{
    double p,n,r;
    cout<<"Enter Principal Amount: ";
    cin>>p;
    cout<<"Enter Interest Rate: ";
    cin>>r;
    cout<<"Enter Number of Periods: ";
    cin>>n;
    double w = Financial::Mortgage::monthlyPayment(r,n,p);
    double x = Financial::Mortgage::totalPayments(w,n);
    cout<<"Total mortgage payment: "<<x<<endl;
    double y = Financial::Retirement::monthlySavings(r,n,p);
    double z = Financial::Retirement::totalSavings(y,n);
    cout<<"Total retirement savings: "<<z<<endl;
}

/*
Enter Principal Amount: 200000
Enter Interest Rate: 0.03
Enter Number of Periods: 30
Total mortgage payment: 192844
Total retirement savings: 180000

Process returned 0 (0x0)   execution time : 13.735 s
Press any key to continue.
```

```
*/
```

Write a program in C++ that creates a class Rectangle with two member variables: width and height.

The class should have a constructor that initializes width and height, and a copy constructor that creates a new object with the same width and height.

Additionally, the class should have a getArea method that returns the area of the rectangle (i.e. width * height).

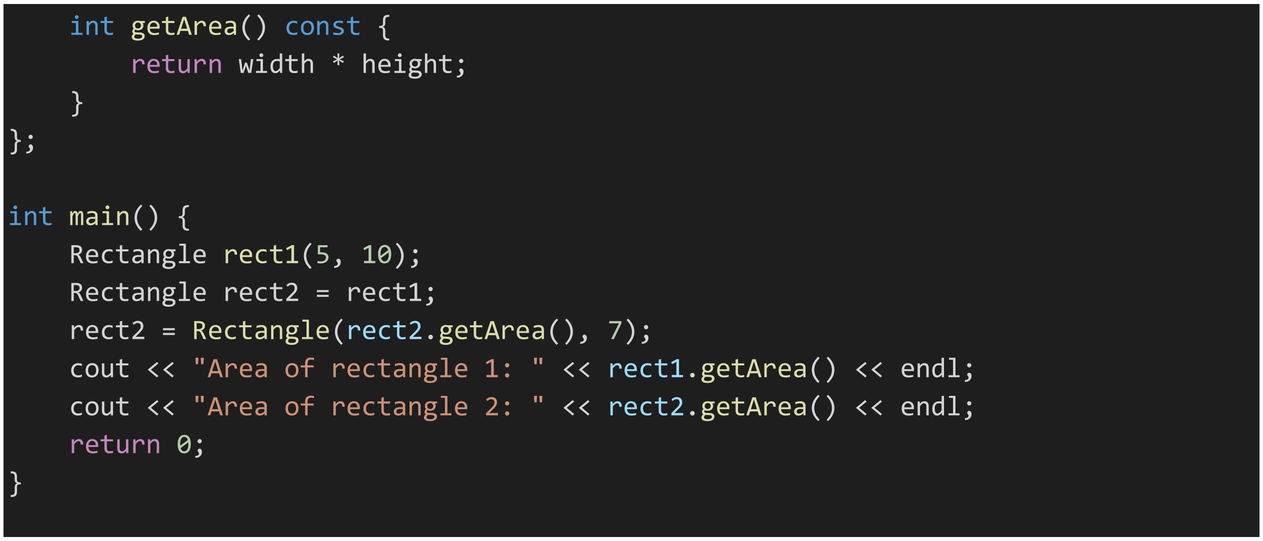
In the main function, create an instance of Rectangle with a width of 5 and a height of 10.

Then, create a new instance of Rectangle using the copy constructor and modify its width to 7.

Finally, print out the area of both rectangles to verify that the original rectangle was not modified by the change in the copy.

Once you have completed the program, you can try extending it further by adding more functionality to the Rectangle class, such as methods for calculating the perimeter or checking if the rectangle is a square.

```cpp
#include <iostream>
using namespace std;

class Rectangle {
private:
    int width, height;

public:
    Rectangle(int w, int h) {
        width = w;
        height = h;
    }
    Rectangle(const Rectangle& r) {
        width = r.width;
        height = r.height;
    }
```



```
    int getArea() const {
        return width * height;
    }
};

int main() {
    Rectangle rect1(5, 10);
    Rectangle rect2 = rect1;
    rect2 = Rectangle(rect2.getArea(), 7);
    cout << "Area of rectangle 1: " << rect1.getArea() << endl;
    cout << "Area of rectangle 2: " << rect2.getArea() << endl;
    return 0;
}
```

Who hasn't heard this iconic quote from Ian Fleming's fictional series James Bond? You will have the pleasure of being a part of his mission today.

The British Secret Service agent is chasing after his archenemy Ernst Stavro Blofeld in Portimão, the southern coast of Portugal. Agent 007 retrieved a piece of harddrive from Blofeld's office last night. Today in the morning, he's meeting his Quartermaster in a museum where he asks for his help to find what's inside the drive. Q discovered that there's a code written in C++ and a documentation of the classes used in the code. Mr Bond asked Q to replace the classes and he'll use this drive as a bait to catch Blofeld.

But before that Q needs to find out the definition of the classes. Here's where you come in. Given the code snippet found in the hard drive and the documentation of the class files, you have to write the definition of the classes for Q. But there's not much time on the clock. You only have 75 mins.

Code Snippet

```
int main(){
Database db;
db.addUser(User("Eva", "green"));
db.addUser(User("Halle", "beryy"));
db.addUser(User("Jane", "seymour"));
db.addUser(User("Rosamund", "pike"));
db.addUser(User("Monica", "belucci"));
db.print();
db.addUser(User("Gemma", "arterton"));
db.addUser(User("Lea", "seydoux"));
User test_user1("Rosamund", "pike");
User test_user2("Rosamund", "berry");
User test_user3("Samantha", "pike");
if(login(db, test_user1))
cout << "Login Successful\n";
else
cout << "Login Failed\n";
if(login(db, test_user2))
cout << "Login Successful\n";
else
cout << "Login Failed\n";
if(login(db, test_user3))
cout << "Login Successful\n";
else
cout << "Login Failed\n";
return 0;
}
```

Documentation of the Classes

```
// include header files
// forward declaration if required

class User {
// name of user as character pointer
// hash of password as long long
public:
// Constructors, Destructors and Copy Constructors as necessary
// In the constructor convert name to lowercase letters and concat
// name after password (password will always be lowercase). Then
// call hash_value function to assign the value of hash variable
long long hash_value(string pass){
const int p = nearest prime number to your student ID % 100
/*
Hash value calculation from pass
if pass is “xyz”
hash = (24 * p * 1) + (25 * p * 2) + (26 * p * 3)
= summation_of(letter_position_in_alphabet * p *
string_position)

Note: Don’t use the ASCII values.
*/
}
// Use friend function / class as necessary
};
class Database{
User * users;
int cur_user;
int max_user;
public:
// Constructors, Destructors and Copy Constructors as necessary
// Whenever a Database class object is created users array will be
// initialized with length 4. One User named “Admin” with password
// “admin” will be created and stored in the first position of the
// users array.

void addUser(User ob){
if(cur_user < max_user){
// add new user in the next available position
}
else {
// double the size of max_user.
// dynamically create a tmp User array of size
// max_user.
memcpy(tmp, users, sizeof(User) * max_user);
// this will copy the contents of users to tmp
// delete old users array
// assign tmp to users (users = tmp)
// Now add new user in the next available position
}
// Don’t forget to increment cur_user
}
void print(){
// Will print the current user count and user names in db.
// See output for formatting
}
// Use friend function / class as necessary
```

```
};

bool login(Database d, User a){

// Iterate through the users in d and if username and hash matches

// the username and hash of a, return true.

// Else return false.

}
```

Output

```
Current User No: 6

Current User's List:

1 Admin

2 Eva

3 Halle

4 Jane

5 Rosamund

6 Monica

Login Successful

Login Failed

Login Failed
```

Marks Distribution

Criteria Marks

Structure of class 4

Constructor, Destructor & Copy Constructor 13

Member and non-member function definition 11

Overall completeness 2

ANSWER:

```
#include <iostream>
#include <cstring>
using namespace std;

class User {
   char* name;
   long long hash;
public:
   User(const char* username, const char* password) {
      int len = strlen(username) + strlen(password) + 1;
      name = new char[len];
      strcpy(name, username);
      strcat(name, password);
      hash = hash_value(password);
   }

   ~User() {
      delete[] name;
   }

   User(const User& other) {
      int len = strlen(other.name) + 1;
      name = new char[len];
      strcpy(name, other.name);
      hash = other.hash;
   }

   long long hash_value(const char* pass) {
      const int p = 13; // Replace with your student ID's nearest prime number % 100
```

```
        long long hash = 0;
        int len = strlen(pass);
        for (int i = 0; i < len; i++) {
            int letter_position = tolower(pass[i]) - 'a' + 1;
            hash += letter_position * p * (i + 1);
        }
        return hash;
    }

    friend class Database;
};

class Database {
    User* users;
    int cur_user;
    int max_user;
public:
    Database() {
        max_user = 4;
        users = new User[max_user];
        cur_user = 0;
        users[cur_user] = User("Admin", "admin");
        cur_user++;
    }

    ~Database() {
        delete[] users;
    }
```

```
Database(const Database& other) {
    max_user = other.max_user;
    cur_user = other.cur_user;
    users = new User[max_user];
    for (int i = 0; i < cur_user; i++) {
        users[i] = other.users[i];
    }
}

void addUser(User ob) {
    if (cur_user < max_user) {
        users[cur_user] = ob;
        cur_user++;
    }
    else {
        max_user *= 2;
        User* tmp = new User[max_user];
        memcpy(tmp, users, sizeof(User) * cur_user);
        delete[] users;
        users = tmp;
        users[cur_user] = ob;
        cur_user++;
    }
}

void print() {
    cout << "Current User No: " << cur_user << endl;
    cout << "Current User's List:" << endl;
    for (int i = 0; i < cur_user; i++) {
            cout << i + 1 << " " << users[i].name << endl;
        }
    }

    friend bool login(Database d, User a);
};

bool login(Database d, User a) {
    for (int i = 0; i < d.cur_user; i++) {
        if (strcmp(d.users[i].name, a.name) == 0 && d.users[i].hash == a.hash) {
            return true;
        }
    }
    return false;
}

int main() {
    Database db;
    db.addUser(User("Eva", "green"));
    db.addUser(User("Halle", "beryy"));
    db.addUser(User("Jane", "seymour"));
    db.addUser(User("Rosamund", "pike"));
    db.addUser(User("Monica", "belucci"));
    db.print();

    db.addUser(User("Gemma", "arterton"));
    db.addUser(User("Lea", "seydoux"));

    User test_user1("Rosamund", "pike");
    User test_user2("Rosamund", "berry");
    User test_user3("Samantha", "pike");

    if (login(db, test_user1))
        cout << "Login Successful" << endl;
    else
        cout << "Login Failed" << endl;

    if (login(db, test_user2))
        cout << "Login Successful" << endl;
    else
        cout << "Login Failed" << endl;

    if (login(db, test_user3))
        cout << "Login Successful" << endl;
    else
        cout << "Login Failed" << endl;

    return 0;
}
```

Consider a small money-transferring system that is implemented using C++. The system consists of two classes, which

are: User and Account: All the attributes in both of the classes are private, on the other hand all the methods in both of

the classes are public. Descriptions of the attributes are illustrated in Table-1, the constructors for both of the classes

are described in Table-2. Description of the methods are illustrated in Table-3.

Class

User Attribute char *id Description Stores a string as the identification number of an user

char Attribute *name Description Stores a string as the name of an user

Class

Table-1

Account User Attribute *user Description Stores the information of an user as the account holder

char Attribute *phone Description Stores a string as the associated phone number of an account

int Attribute *balance Description Stores a single integer as the balance of an account

Table-2

Class User Constructor User(char*, char*); Description Initializes user->id with the first parameter and user->name with the second parameter

Class Account Constructor Account(char*, char*, char*); Description Initializes user with the first two parameters and the phone with the third parameter. Sets the value of balance to 500.

Table-3

Class Method Description

User ~User( ); Deallocates the allocated memory Account

void setId(char *x); Sets the value of user->id to x

void setName(char *x); Sets the value of user->name to x

void setPhone(char *x); Sets the value of phone to x

void print( ); Prints the information of an account. For example: Let the user->id,

user->name, phone, balance of an Account are respectively (“S-11”, “Zarif”, “01211997711”, 200) then the method will print these information in the following format:

ID: S-11, Name: Zarif, Phone : 01211997711, Balance: 200

~Account( ); Deallocates the allocated memory

In the global scope create an array of objects of Account class of size 4 named acc. Initialize the objects of Account

with the following values:

{("10", "Rafi", "010"), ("15", "Binita", "015"), ("29", "Nabil", "029"), (“36", "Maisha", "036")};

There is also a global function named transferMoney that takes three integers (i, j, amount) as parameters. It

represents that acc[i] is the sender and acc[j] is the receiver. If the balance of sender is greater or equal to the amount

then that amount is reduced from the balance of sender and added to the balance of receiver and prints a message

“Transaction successful” otherwise prints “Insufficient balance”. Also if the phone of both Accounts are the same then

the function does nothing which means the transaction has failed and prints a message “Invalid transaction”.

Special Instruction

● Both of the classes do not have any other constructors except the constructors described in Table-2. But the

copy constructors may be overwritten if required.

● It is allowed to define any function or class as friend of any class

● Implementing any other additional methods for any class is not allowed

● Using the header <bits/stdc++.h> is prohibited

Tasks

● Implement the two classes and the global function

● Create an Account in the main function named admin which will have the same set of values as acc[0]. Note

that acc[0] and admin are two different entities though their attributes are being initialized with the same set of

values. Later the admin may update its attributes but that will not update anything in acc[0].

● Update the attributes of admin as the following

user->id: “A-11”

user->name: “Admin”

phone: “01511”

● Do the following operations:

```
transfer(2, 0, 100);

transfer(1, 0, 150);

transfer(1, 1, 100);

transfer(2, 3, 800);
```

● Now print all the elements of the acc array and then print the admin also. Your output should look like as the

following:

```
Transaction successful

Transaction successful

Invalid transaction

Insufficient balance

ID: 10, Name: Rafi, Phone: 010, Balance: 750

ID: 15, Name: Binita, Phone: 015, Balance: 350

ID: 29, Name: Nabil, Phone: 029, Balance: 400

ID: 36, Name: Maisha, Phone: 036, Balance: 500

ID: A-11, Name: Admin, Phone: 01511, Balance: 500
```

ANSWER:

```
#include <iostream>

#include <cstring>

using namespace std;


class User {

  char* id;

  char* name;

public:

  User(char* uid, char* uname) {

    int len_id = strlen(uid) + 1;
```

```
        id = new char[len_id];
        strcpy(id, uid);

        int len_name = strlen(uname) + 1;
        name = new char[len_name];
        strcpy(name, uname);
    }

    ~User() {
        delete[] id;
        delete[] name;
    }

    void setId(char* x) {
        delete[] id;
        int len_id = strlen(x) + 1;
        id = new char[len_id];
        strcpy(id, x);
    }

    void setName(char* x) {
        delete[] name;
        int len_name = strlen(x) + 1;
        name = new char[len_name];
        strcpy(name, x);
    }

    friend class Account;
};
```

```
class Account {
    User* user;
    char* phone;
    int balance;
public:
    Account(char* uid, char* uname, char* ph) {
        user = new User(uid, uname);
        int len_phone = strlen(ph) + 1;
        phone = new char[len_phone];
        strcpy(phone, ph);
        balance = 500;
    }

    ~Account() {
        delete user;
        delete[] phone;
    }

    void setPhone(char* x) {
        delete[] phone;
        int len_phone = strlen(x) + 1;
        phone = new char[len_phone];
        strcpy(phone, x);
    }

    void print() {
        cout << "ID: " << user->id << ", Name: " << user->name << ", Phone: " << phone << ", Balance: " << balance << endl;
    }

    friend void transferMoney(Account& sender, Account& receiver, int amount);
};

void transferMoney(Account& sender, Account& receiver, int amount) {
    if (strcmp(sender.phone, receiver.phone) == 0) {
        cout << "Invalid transaction" << endl;
        return;
    }

    if (sender.balance >= amount) {
        sender.balance -= amount;
        receiver.balance += amount;
        cout << "Transaction successful" << endl;
    } else {
        cout << "Insufficient balance" << endl;
    }
}

int main() {
    Account acc[4] = {
        Account("10", "Rafi", "010"),
        Account("15", "Binita", "015"),
        Account("29", "Nabil", "029"),
        Account("36", "Maisha", "036")
    };

    Account admin("10", "Rafi", "010");
    admin.user->setId("A-11");
    admin.user->setName("Admin");
    admin.setPhone("01511");

    transferMoney(acc[2], acc[0], 100);
    transferMoney(acc[1], acc[0], 150);
    transferMoney(acc[1], acc[1], 100);
    transferMoney(acc[2], acc[3], 800);

    for (int i = 0; i < 4; i++) {
        acc[i].print();
    }
    admin.print();

    return 0;
}
```

Question - 2

“Even 5-month-old infants can calculate the results of simple arithmetical operations on small numbers of items. This

indicates that infants possess true numerical concepts, and suggests that humans are innately endowed with

arithmetical abilities.”

[Karen Wynn, Addition and subtraction by human infants, Nature 1992]

In this problem, all you have to do is the job of an infant; ADDITION: a trivial arithmetic operation. Design a class

named Operation, which can handle two data values, where the values can be of any data types. Next, write a Member

function named Addition. The function shall be able to add the input values (Lets say C1 and C2) and output the

resultant value, depending on the data types specified by the user.

The input of the program starts with an integer, indicating the number of test cases. Then, the program will take 2

inputs; first indicates the data type of the resultant value, while the second indicates the data type of the input values

(C1 and C2).

Sample Input

3

int float

3 3.25

float int

3.65 3.78

string string

NO FIRE

Sample Output

Test Case 1: 6,

Test Case 2: 6,

Test Case 3: NOFIRE

```
#include <iostream>
#include <string>
using namespace std;

class Operation {
public:
    template <typename T1, typename T2>
    static void Addition(T1 val1, T2 val2) {
        cout << "Test Case: ";
        cout << val1 + val2 << endl;
    }
};

int main() {
    int numCases;
    cin >> numCases;

    for (int i = 0; i < numCases; i++) {
        string resultType, val1Type;
        cin >> resultType >> val1Type;

        if (resultType == "int" && val1Type == "int") {
            int val1, val2;
            cin >> val1 >> val2;
            Operation::Addition(val1, val2);
        } else if (resultType == "float" && val1Type == "float") {
            float val1, val2;
            cin >> val1 >> val2;
            Operation::Addition(val1, val2);
        } else if (resultType == "float" && val1Type == "int") {
            float val1;
            int val2;
            cin >> val1 >> val2;
            Operation::Addition(val1, static_cast<float>(val2));
        } else if (resultType == "int" && val1Type == "float") {
            int val1;
            float val2;
            cin >> val1 >> val2;
            Operation::Addition(static_cast<float>(val1), val2);
        } else if (resultType == "string" && val1Type == "string") {
            string val1, val2;
            cin >> val1 >> val2;
            Operation::Addition(val1, val2);
        }
    }

    return 0;
}
```

Gringotts Wizarding Bank is the only bank of the wizarding world, and is owned and operated by goblins. According to Rubeus Hagrid, other than Hogwarts School of Witchcraft and Wizardry, Gringotts is the safest place in the wizarding world. In addition to storing money and valuables for wizards and witches, one can go there to exchange muggle money for wizarding money.

You, being very curious about Gringotts, want to figure out how the interest scheme works for wizards and goblins. Because goblins also keep their valuables in the vaults of Gringotts. Design two classes "Goblin (Bankers) and Customers with necessary attributes and methods. The classes should contain the following attributes.

Goblin

name - char * store - names of Goblin.

designation string Goblins can be assigned with any of the 3

designations

1. manager - definitely has a personal vault atGringotts.
2. accountant - also has a personal vault atGringotts.
3. coiner (equivalent to cashier of a mugglebank) - these Goblins don't have anypersonal vaults.

salary double Managers - 1000 gold coins

Accountant - 600 gold coins

Coiners - 300 gold coins

object Customer * If the Goblin has a vault, his customer details will bestored in this object. Otherwise assign null.

display(year) void displays name, designation and total income in

specified year(s) of a Goblin.

total_income(year) double if no vault at Gringotts:

income = salary * year

if has a vault at Gringotts:

income = salary * year + interest(year)

Customer

name char * store names of Customer.

vault_no integer A unique number that indicates the vault assigned

to the customer.

rank integer indicates the rank of the customer. Can be of rank 1,

rank 2 or rank 3.

interest_rate double 7% = rank 1

5% = rank 2

3.5% = rank 3.

vault_balance double Initial deposit of the customer in Gringotts.

total_balance(year) double balance = vault_balance * (1 + interest_rate * year)

interest(year) double [interest = total_balance(year) - vault_balance

display(year) void displays name, vault_no and total balance in

specified year(s) of a Customer.

find_goblin(Goblin [], name,size)

```
find_customer(Customer [],name, size)
```

Sample Input and Output:

Customer and Goblin Details Input Menu Driven Output

```
Enter number of Customers: 5
Enter Details:
Name: Albus Dumbledore
Rank: 1
Vault Balance: 10000
Name: Blordak
Rank: 2
Vault Balance: 200
Name: Harry Potter
Rank: 3
Vault Balance: 5000
Name: Bellatrix Lestrange
Rank: 2
Vault Balance: 3500
Name: Griphook
Rank: 1
Vault Balance: 4800
Enter number of Goblins: 3
Enter Details:
Name: Griphook
Designation: Manager
Name: Blordak
Designation: Accountant
Name: Snaglok
Designation: Coiner
```

```
See details of a Goblin/Customer: G
Enter Name: Blordak
Enter Year: 3
Name: Blordak
Designation: Accountant
Total Income: 1830 Gold Coins
See details of a Goblin/Customer: C
Enter Name: Bellatrix Lestrange
Enter Year: 5
Name: Bellatrix Lestrange
Valut No: 912
Total Balance: 3850 Gold Coins
See details of a Goblin/Customer: C
Enter Name: Griphook
Enter Year: 2
Name: Griphook
Valut No: 666
Total 0.Balance: 5472 Gold Coins
See details of a Goblin/Customer: G
Enter Name: Griphook
Enter Year: 2
Name: Griphook
Designation: Manager
Total Income: 2672 Gold Coins
```

Notes

1. Names of the Goblins and Customers will be unique and not separated by space.

2. If you use rand() to generate vault_no, use srand(time(0)) to set the seed.

3. Don’t write any set() and get() methods in th”e class.

4. Write proper constructors and destructors where necessary.

5. You may use strcmp() to compare char arrays.

ANSWER:

```
#include <iostream>

#include <cstring>

#include <ctime>

#include <cstdlib>

using namespace std;


class Customer; // Forward declaration of the Customer class for Goblin to use.


class Goblin {

    char* name;

    char* designation;

    double salary;

    Customer* customer; // If Goblin has a vault, this will point to the corresponding Customer object.


public:

    Goblin(char* name, char* designation);

    ~Goblin();

    void display(int year);

    double total_income(int year);

    void set_customer(Customer* customer);


    static double get_interest_rate(const int rank);

};


class Customer {
    char* name;
    int vault_no;
    int rank;
    double interest_rate;
    double vault_balance;

public:
    Customer(char* name, int rank, double vault_balance);
    ~Customer();
    double total_balance(int year);
    double interest(int year);
    void display(int year);

    int get_vault_no() const;
};

Goblin::Goblin(char* name, char* designation) {
    this->name = new char[strlen(name) + 1];
    strcpy(this->name, name);

    this->designation = new char[strlen(designation) + 1];
    strcpy(this->designation, designation);

    if (strcmp(designation, "manager") == 0) {
        salary = 1000;
    } else if (strcmp(designation, "accountant") == 0) {
        salary = 600;
    } else if (strcmp(designation, "coiner") == 0) {
        salary = 300;
    }

    customer = nullptr;
}

Goblin::~Goblin() {
    delete[] name;
    delete[] designation;
}

void Goblin::display(int year) {
    cout << "Name: " << name << endl;
    cout << "Designation: " << designation << endl;
    cout << "Total Income: " << total_income(year) << " Gold Coins" << endl;
}

double Goblin::total_income(int year) {
    double income = salary * year;
    if (customer) {
        income += customer->interest(year);
    }
    return income;
}

void Goblin::set_customer(Customer* customer) {
    this->customer = customer;
}

double Goblin::get_interest_rate(const int rank) {
    if (rank == 1) {
        return 0.07;
    } else if (rank == 2) {
        return 0.05;
    } else if (rank == 3) {
        return 0.035;
    }
    return 0.0;
}

Customer::Customer(char* name, int rank, double vault_balance) {
    this->name = new char[strlen(name) + 1];
    strcpy(this->name, name);

    srand(time(0)); // Initialize random seed for generating vault_no
    vault_no = rand() % 900 + 100; // Generate a random 3-digit vault number

    this->rank = rank;
    this->vault_balance = vault_balance;

    interest_rate = get_interest_rate(rank);
}

Customer::~Customer() {
    delete[] name;
}

double Customer::total_balance(int year) {
    return vault_balance * (1 + interest_rate * year);
```

```
}

double Customer::interest(int year) {
    return total_balance(year) - vault_balance;
}

int Customer::get_vault_no() const {
    return vault_no;
}

void Customer::display(int year) {
    cout << "Name: " << name << endl;
    cout << "Valut No: " << vault_no << endl;
    cout << "Total Balance: " << total_balance(year) << " Gold Coins" << endl;
}

int main() {
    int numCustomers;
    cout << "Enter number of Customers: ";
    cin >> numCustomers;

    Customer* customers[numCustomers];

    cout << "Enter Details:" << endl;
    for (int i = 0; i < numCustomers; i++) {
        char name[100];
        int rank;
        double vault_balance;
        cout << "Name: ";
        cin.ignore(); // Ignore any newline character left in the input buffer
        cin.getline(name, 100);

        cout << "Rank: ";
        cin >> rank;

        cout << "Vault Balance: ";
        cin >> vault_balance;

        customers[i] = new Customer(name, rank, vault_balance);
    }

    int numGoblins;
    cout << "Enter number of Goblins: ";
    cin >> numGoblins;

    Goblin* goblins[numGoblins];

    cout << "Enter Details:" << endl;
    for (int i = 0; i < numGoblins; i++) {
        char name[100];
        char designation[20];

        cout << "Name: ";
        cin.ignore(); // Ignore any newline character left in the input buffer
        cin.getline(name, 100);

        cout << "Designation: ";
        cin.getline(designation, 20);

        goblins[i] = new Goblin(name, designation);
    }

    char choice;
    do {
        cout << "See details of a Goblin/Customer (G/C): ";
        cin >> choice;

        if (choice == 'G' || choice == 'g') {
            char name[100];
            int year;

            cout << "Enter Name: ";
            cin.ignore(); // Ignore any newline character left in the input buffer
            cin.getline(name, 100);

            cout << "Enter Year: ";
            cin >> year;

            bool found = false;
            for (int i = 0; i < numGoblins; i++) {
                if (strcmp(goblins[i]->name, name) == 0) {
                    goblins[i]->display(year);
                    found = true;
                    break;
                }
            }
            if (!found) {
                cout << "Goblin with name " << name << " not found." << endl;
            }
        } else if (choice == 'C' || choice == 'c') {
            char name[100];
            int year;

            cout << "Enter Name: ";
            cin.ignore(); // Ignore any newline character left in the input buffer
            cin.getline(name, 100);

            cout << "Enter Year: ";
            cin >> year;

            bool found = false;
            for (int i = 0; i < numCustomers; i++) {
                if (strcmp(customers[i]->name, name) == 0) {
                    customers[i]->display(year);
                    found = true;
                    break;
                }
            }

            if (!found) {
                cout << "Customer with name " << name << " not found." << endl;
            }
        } else {
            cout << "Invalid choice. Please enter G or C." << endl;
```

```
        }

        cout << "Continue? (Y/N): ";
        cin >> choice;
    } while (choice == 'Y' || choice == 'y');

    // Clean up memory
    for (int i = 0; i < numCustomers; i++) {
        delete customers[i];
    }

    for (int i = 0; i < numGoblins; i++) {
        delete goblins[i];
    }

    return 0;
}
```

Create a class called Time that has separate private member data for hours, minutes, and seconds. You have another member function to calculate the time difference. You set time with the local time of the different cities of different countries. When you are given two local times, calculate the time difference.

Sample Input

City 1 Name: Dhaka

Time of Dhaka: 11:00:20

City 2 Name: California

Time of California: 22:10:30

Sample Output

Time difference between Dhaka and

California: 11 hrs 10 minutes and 10 seconds

ANSWER:

```
#include<iostream>

#include<cmath>

#include<string>

using namespace std;

class Time{

   int hour,minute,second;

public:

   Time(){

      hour=0;

      minute=0;

      second=0;

   }

   Time(int hour, int minute, int second){

      this->hour=hour;

      this->minute=minute;

      this->second=second;

   }

   void print(){

      cout<<endl<<hour<<" "<<minute<<" "<<second<<endl;

   }

   friend void difference(Time t1,Time t2);// friend void difference(Time,Time); will also be valid //diffe is my friend and so access my members to the function

};

void difference(Time t1,Time t2){  //usage of friend function and friend class(well make difference function a friend of time class)
```

```
     int hr=abs(t1.hour-t2.hour);

      int min_=abs(t1.minute-t2.minute);

      int sec=abs(t1.second-t2.second);

      cout<<hr<<" "<<min_<<" "<<sec<<endl;

   }

int main(){

   string s1,s2;


   cout<<"City 1 name: ";

   cin>>s1;

   cout<<endl;

   cout<<"Time of "<<s1<<": ";

   int h,m,s;

   char p;

   cin>>h>>p>>m>>p>>s;

   Time t1(h,m,s);//cout<<endl<<h<<" "<<m<<" "<<s<<endl;

   t1.print();

   cout<<endl;


   cout<<endl<<"City 2 name: ";

   cin>>s2;

   cout<<endl;

   cout<<"Time of "<<s2<<": ";

   cin>>h>>p>>m>>p>>s;

   Time t2(h,m,s);//cout<<endl<<h<<" "<<m<<" "<<s<<endl;

   t2.print();

   cout<<endl;

   //Time t(11,10,10);

   //t.print();
```

```
    difference(t1,t2);
    retuen 0;
}
```

· Define two namespaces, "performance" and "salary," each containing a class with the same name, "Employee."

· Private attributes of Employee class under the “performance” namespace are id and performance_rating (range 1-5)

· Employee class under “salary” namespace has a private attribute named sal.

· Take an array (size=5) of object for Employee class and then take performance rantings as input and set them against the corresponding id.

· Now check the salary of an id based on performance_rating. Get the performance_rating from the Employee class under the “performance” namespace and then pass it to the function of the Employee class under the “salary” namespace to get the salary.

Salary Criteria

performance_rating>=4: salary=40000.

performance_rating=3: salary=30000.

performance_rating=2: salary=25000.

performance_rating=1: salary=20000.

Sample Input

```
Enter the Performance of employee
(rating from 1-5)
Performance for id 0: 5
Performance for id 1: 1
Performance for id 2: 2
Performance for id 3: 4
Performance for id 4: 3
```

```
Enter the id for salary :3
```

Sample Output

```
Salary of id 3 : 40000
```

ANSWEWR:

```
#include <iostream>

namespace performance {
    class Employee {
    private:
        int id;
        int performance_rating;

    public:
        Employee(int id) : id(id), performance_rating(0) {}

        void setPerformance(int rating) {
            if (rating >= 1 && rating <= 5)
                performance_rating = rating;
            else
                std::cout << "Invalid performance rating. It should be in the range 1-5." << std::endl;
        }

        int getPerformance() const {
            return performance_rating;
        }
    };
} // namespace performance

namespace salary {
  class Employee {
  private:
    int sal;

  public:
    Employee() : sal(0) {}

    int calculateSalary(int performance_rating) {
      if (performance_rating >= 4)
        sal = 40000;
      else if (performance_rating == 3)
        sal = 30000;
      else if (performance_rating == 2)
        sal = 25000;
      else if (performance_rating == 1)
        sal = 20000;

      return sal;
    }
  };
} // namespace salary

int main() {
  performance::Employee employees[5] = {0, 1, 2, 3, 4};
  int rating;

  std::cout << "Enter the Performance of employee (rating from 1-5)" << std::endl;

  for (int i = 0; i < 5; i++) {
    std::cout << "Performance for id " << i << ": ";
    std::cin >> rating;
    employees[i].setPerformance(rating);
  }

  int id;
  std::cout << "Enter the id for salary: ";
  std::cin >> id;

  int performance_rating = employees[id].getPerformance();

  salary::Employee salary_employee;
  int salary = salary_employee.calculateSalary(performance_rating);

  std::cout << "Salary of id " << id << ": " << salary << std::endl;

  return 0;
}
```

Military Institute of Science & Technology

Department of Computer Science and Engineering

CSE 206: Object Oriented Programming Language Sessional

Practice Problem Compilation

/*

FileHandler Class - Problem on Copy Constructor.

Our task is to design a class that handles the file opening and closing for us.

a. Find and print the number of vowels in a given text file.

b. Print the content of the text file.

Write down a class named FileHandler. It will have the following skeleton.

```
class FileHandler {
FILE * fp;
char name[100];
public:
};
```

The constructor will receive the filename and initialize the file pointer as well as the name (of the file). The destructor will close the file.

Sample Code for opening a file:

```
FILE * fp = fopen(�input.txt�, �r�);
```

Sample Code for closing a file:

```
fclose(fp);
```

The class will have two friend functions.

a. void printContent(FileHandler fh)

Prints the content of the file pointed by the filepointer in fh.

b. void printVowelCount(FileHandler fh)

Prints the total number of vowels in the file.

Sample code for reading from a file:

```
char line[100]; //Let, each line-length <100
```

```
while(!feof(fp)) {
fgets(line, 100, fp); //read one line
printf("%s", line); //print the line
}
*/
#include<iostream>
#include <cstdio>
#include <cstring>
#include <cctype>
using namespace std;
class FileHandler {
private:
    FILE * fp;
    char name[100];
public:
    FileHandler(const char * filename) : fp(NULL) {
        fp = fopen(filename, "r");
        if (fp == NULL) {
            printf("Error: cannot open file '%s'\n", filename);
        }
        strncpy(name, filename, 100);
    }

    ~FileHandler() {
        if (fp != NULL) {
            fclose(fp);
        }
    }
    friend void printContent(FileHandler fh);
    friend void printVowelCount(FileHandler fh);
};

void printContent(FileHandler fh) {
    FILE * fp = fh.fp;
    if (fp == NULL) {
        printf("Error: file not opened.\n");
        return;
    }
    char line[100];
    while (fgets(line, 100, fp) != NULL) {
        printf("%s", line);
    }
}

void printVowelCount(FileHandler fh) {
    FILE * fp = fh.fp;
    if (fp == NULL) {
        printf("Error: file not opened.\n");
        return;
    }
    char line[100];
    int count = 0;
    while (fgets(line, 100, fp) != NULL) {
        for (int i = 0; line[i] != '\0'; i++) {
            char c = tolower(line[i]);
            if (c == 'a' || c == 'e' || c == 'i' || c == 'o' || c == 'u') {
                count++;
            }
        }
    }
    printf("Total number of vowels in '%s': %d\n", fh.name, count);
}

int main() {
    FileHandler fh("input.txt");
    printContent(fh);
    printVowelCount(fh);
    return 0;
}

--------------------------------------------------------------------------------------------------------------------------------

/*
Consider the following skeleton for problem a, b and c.
class String
{
int len;
char *ptr;
public: //declare constructor, destructor and copy constructor
char get(int ind)

{
//return the character at the given index
//Check if index is out of bound
//return 0 in case of error
}
```

```
int put(int ind,char c)
{
//Assign character c at position index
//Check if index is out of bound
//return -1 in case of error
}
int getlength()
{
//return the allocation length of the string
}
void print()
{
//print the string upto allocation size
}


};
```

a. Design a String class that will hold the character values dynamically. Use the above mentioned skeleton. Declare a non-member function, which will compare the sum of ASCII characters between two strings. If the sum of ASCII characters of String 1 is greater than String 2, �1� will be printed whereas, if the sum of ASCII characters of String 2 is greater than String 1, �2� will be printed. However, for the sum of ASCII characters of two strings being the same, 0 will be printed by the program.

```
void compare(String s1,String s2)
{
// print 1; if sum of ASCII character of s1 is greater than s2
// print 2; if sum of ASCII character of s2 is greater than s1
// print 0; if sum of ASCII characters of both string being
the same.
```

```
}
b. Declare a non-member function of the class of problem 1, which will join 2
strings and will return the resultant string.
String concat(String s1, String s2){
//string s1=�abcd�
//string 2 = �1234�
//result= = �abcd1234�
//Joins two strings s1 and s2 and returns the resultant
string.
}

c. Declare a non-member function of the class of problem 1, which will insert
a particular character at a fixed position of the string and other characters
of the string will be shifted to the right
void insert(String &st, int index, char c) {
//string s1=�abcd�
//insert(s1,0,�k�) //kabc
// Insert char c at index position of st. Shift other characters to
the right.}
*/

#include <iostream>
#include <cstring>

using namespace std;

class String {
  public:
  int len;
```

```
char *ptr;
// constructor
String() {
   len = 0;
   ptr = new char[len+1];
   ptr[0] = '\0';
}

// parameterized constructor
String(const char *s) {
   len = strlen(s);
   ptr = new char[len+1];
   strcpy(ptr, s);
}

// destructor
~String() {
   delete[] ptr;
}

// copy constructor
String(const String &s) {
   len = s.len;
   ptr = new char[len+1];
   strcpy(ptr, s.ptr);
}

// get character at index
char get(int ind) {
        if (ind < 0 || ind >= len) {
            return 0;
        }
        return ptr[ind];
    }

    // put character at index
    int put(int ind, char c) {
        if (ind < 0 || ind >= len) {
            return -1;
        }
        ptr[ind] = c;
        return 0;
    }

    // get length of string
    int getlength() {
        return len;
    }

    // print string
    void print() {
        cout << ptr << endl;
    }
};

// compare two strings by sum of ASCII characters
void compare(String s1, String s2) {
    int sum1 = 0, sum2 = 0;
    for (int i = 0; i < s1.getlength(); i++) {
        sum1 += (int)s1.get(i);
    }
    for (int i = 0; i < s2.getlength(); i++) {
        sum2 += (int)s2.get(i);
    }
    if (sum1 > sum2) {
        cout << "1" << endl;
    } else if (sum2 > sum1) {
        cout << "2" << endl;
    } else {
        cout << "0" << endl;
    }
}

// concatenate two strings
String concat(String s1, String s2) {
    int len1 = s1.getlength();
    int len2 = s2.getlength();
    char *newptr = new char[len1+len2+1];
    strcpy(newptr, s1.ptr);
    strcat(newptr, s2.ptr);
    String result(newptr);
    delete[] newptr;
    return result;
}

// insert character at a fixed position in string
void insert_(String &st, int index, char c) {
    int len = st.getlength();
    char *newptr = new char[len+2];
    for (int i = 0; i <= len; i++) {
        if (i < index) {
            newptr[i] = st.get(i);
        } else if (i == index) {
            newptr[i] = c;
        } else {
            newptr[i] = st.get(i-1);
        }
    }
    st = String(newptr);
    delete[] newptr;
}

int main() {
    // testing String class and non-member functions
    String s1("hello");
    String s2("world");
    String s3 = concat(s1, s2);
    s3.print();
    compare(s1, s2);
    insert_(s3, 5, '-');
    s3.print();
    return 0;
}
```

```
/*
In this problem, you�ll need to design a Point class that will hold the Cartesian
coordinates. Your class will have two private variables named x and y.
a. Write down appropriate constructors and destructors.
b. Write a member function with the following prototype:
Point * shiftBy(int dx, int dy)
[Suppose a point object is p1(3, 4). Calling p1.shiftBy(2, 6)will shift p1 to
(5, 10), and return itself.]
c. Write a non-member function to calculate the distance between two
functions. It will have the following prototype.
double distance(Point p1, Point p2)
d. Write a print() function to print the coordinates.
Requirement
a. Declare a Point object p1 with values
b. Declare a pointer to Point object named p2.
c. Shift p1 by (2, 5), and hold its reference in p2.
d. Use p2 pointer to print the value of p1
*/
#include <iostream>
#include <cmath>
using namespace std;
class Point {
private:
    int x, y;
public:
    // Constructors
    Point() {
        x = 0;
        y = 0;
    }

    Point(int x, int y) {
        this->x = x;
        this->y = y;
    }

    // Destructor
    ~Point() {}

    // Member function to shift point by dx and dy
    Point * shiftBy(int dx, int dy) {
        x += dx;
        y += dy;
        return this;
    }

    // Non-member function to calculate distance between two points
    friend double distance(Point p1, Point p2) {
        int dx = p1.x - p2.x;
        int dy = p1.y - p2.y;
        return std::sqrt(dx*dx + dy*dy);
    }

    // Print function to print coordinates
    void print() {
        std::cout << "(" << x << ", " << y << ")" << std::endl;
    }
};
```

```
int main() {
   // Declare a Point object p1 with values
   Point p1(3, 4);

   // Declare a pointer to Point object named p2
   Point *p2;

   // Shift p1 by (2, 5), and hold its reference in p2
   p2 = p1.shiftBy(2, 5);

   // Use p2 pointer to print the value of p1
   p2->print();

   return 0;
}
```

```
/*
In this problem, you�ll need to design a Stack class that will hold the integer values
dynamically. Use the following skeleton of Stack class.
class Stack
{
int *p; int len;
public:
Stack(int n);//n = size of the stack
~Stack(){delete [ ]p;}
initStack(int *ara)//init the stack with the values of ara
printStack()//print the values of stack
...//Write more methods if necessary
```

```
};
void printSum(Stack s)//non-member function, prints the sum of
elements of s.
*/
#include <iostream>

using namespace std;

class Stack {
    int *p;
    int len;
public:
    Stack(int n) {
        p = new int[n];
        len = n;
    }

    void initStack(int *ara) {
        for (int i = 0; i < len; i++) {
            p[i] = ara[i];
        }
    }

    void printStack() {
        for (int i = 0; i < len; i++) {
            cout << p[i] << " ";
        }
        cout << endl;
    }
    void printSum(Stack s) {
    int sum = 0;
    for (int i = 0; i < s.len; i++) {
        sum += s.p[i];
    }
    cout << "Sum of stack elements: " << sum << endl;
}


    ~Stack() {
        delete[] p;
    }
};


int main() {
    int ara[] = {1, 2, 3, 4, 5};
    int n = sizeof(ara) / sizeof(ara[0]);

    Stack s(n);
    s.initStack(ara);
    s.printStack();
    s.printSum(s);

    return 0;
}
```

```
/*
A complex variable has a real portion and an imaginary portion.
a. Write down a class that represents a complex variable (all the members will
be private)
b. Write a public setter method to set the values.
c. Write a non-member function add(...) that will take two complex Variable
objects and print their summation
d. Write a non-member function sub(...) that will take two complex Variable
objects and print their subtraction.
*/
#include <iostream>
using namespace std;
class Complex {
private:
  double real;
  double imaginary;
public:
  Complex() : real(0), imaginary(0) {}
  Complex(double r, double i) : real(r), imaginary(i) {}
  void setComplex(double r, double i) {
    real = r;
    imaginary = i;
  }
  friend void add(Complex c1, Complex c2);
  friend void sub(Complex c1, Complex c2);
};

void add(Complex c1, Complex c2) {
  double real = c1.real + c2.real;
  double imaginary = c1.imaginary + c2.imaginary;
  cout << "Sum: " << real << " + " << imaginary << "i" << endl;
}
```

```
void sub(Complex c1, Complex c2) {
   double real = c1.real - c2.real;
   double imaginary = c1.imaginary - c2.imaginary;
   cout << "Difference: " << real << " + " << imaginary << "i" << endl;
}

int main() {
   Complex c1, c2;
   c1.setComplex(3.0, 4.0);
   c2.setComplex(1.0, 2.0);
   add(c1, c2);
   sub(c1, c2);
   return 0;
}
```

```
/*
A Science Student of SSC has a roll number, marks in primary Subjects, and marks
in Science subjects.
A Commerce Student has a roll number, marks in primary subjects, and marks in
Commerce Subjects.
a. Write down two classes with the mentioned properties. (all private)
b. Write necessary public setter functions
c. Write a non-member compare(�) function that will take a science student
object and a commerce student object, and return the roll number of the
student having higher average overall marks.
*/
#include <iostream>
using namespace std;
```

```
class ScienceStudent {
private:
    int rollNo;
    int primaryMarks;
    int scienceMarks;
public:
    void setRollNo(int rollNo) {
        this->rollNo = rollNo;
    }
    void setPrimaryMarks(int primaryMarks) {
        this->primaryMarks = primaryMarks;
    }
    void setScienceMarks(int scienceMarks) {
        this->scienceMarks = scienceMarks;
    }
    float getAverageMarks() {
        return (primaryMarks + scienceMarks) / 2.0;
    }
    int getRollNo() {
        return rollNo;
    }
};

class CommerceStudent {
private:
    int rollNo;
    int primaryMarks;
    int commerceMarks;
public:
    void setRollNo(int rollNo) {
        this->rollNo = rollNo;
    }
    void setPrimaryMarks(int primaryMarks) {
        this->primaryMarks = primaryMarks;
    }
    void setCommerceMarks(int commerceMarks) {
        this->commerceMarks = commerceMarks;
    }
    float getAverageMarks() {
        return (primaryMarks + commerceMarks) / 2.0;
    }
    int getRollNo() {
        return rollNo;
    }
};

int compare(ScienceStudent science, CommerceStudent commerce) {
    if (science.getAverageMarks() > commerce.getAverageMarks()) {
        return science.getRollNo();
    } else {
        return commerce.getRollNo();
    }
}

int main() {
    ScienceStudent science;
    CommerceStudent commerce;
    science.setRollNo(101);
    science.setPrimaryMarks(80);
    science.setScienceMarks(90);

    commerce.setRollNo(201);
    commerce.setPrimaryMarks(70);
    commerce.setCommerceMarks(85);

    int rollNo = compare(science, commerce);
    cout << "Student with higher average marks has roll number: " << rollNo << endl;

    return 0;
}
```

```
/*
Solve the problem using Object Oriented Paradigm.
Given an integer n, find the value of
a. 1 + 2 + 3 + � � � + n
b. 1^2 + 2^2 + 3^2 + � � � + n^2
c. 1^3 + 2^3 + 3^3 + � � � + n^3
*/
#include <iostream>
using namespace std;

class Summation {
private:
    int n;
public:
    Summation(int n) {
        this->n = n;
    }
    int sum() {
        int s = 0;
        for (int i = 1; i <= n; i++) {
            s += i;
        }
        return s;
    }
    int sumSquares() {
        int s = 0;
        for (int i = 1; i <= n; i++) {
            s += i * i;
        }
        return s;
    }
    int sumCubes() {
        int s = 0;
        for (int i = 1; i <= n; i++) {
            s += i * i * i;
        }
        return s;
    }
};

int main() {
    int n;
    cout << "Enter the value of n: ";
    cin >> n;
    Summation s(n);
    cout << "Summation of 1 to n: " << s.sum() << endl;
    cout << "Summation of squares of 1 to n: " << s.sumSquares() << endl;
    cout << "Summation of cubes of 1 to n: " << s.sumCubes() << endl;
    return 0;
}
```

```
/*
Solve the problem using Object Oriented Paradigm.
A library has some books on Data Structure and some books on Algorithms.
- Books can be issued (taken from library)
- Books can be returned (to the library)
- If book-count falls below 2, a warning is shown
- If book-count is zero, it cannot be issued from the library
*/
#include <iostream>
#include <string>
using namespace std;
class Book {
    private:
        string title;
        int bookCount;

    public:
        Book(string title, int bookCount): title(title), bookCount(bookCount) {}

        void issue() {
            if (bookCount > 0) {
                bookCount--;
                if (bookCount < 2) {
                    cout << "Warning: Only " << bookCount << " copies of " << title << " left." << endl;
                }
                cout << "Issued " << title << "." << endl;
            }
            else {
                cout << "Sorry, " << title << " is not available." << endl;
            }
        }

        void returnBook() {
            bookCount++;
            cout << "Returned " << title << "." << endl;
        }
};

int main() {
    Book dsBook("Data Structures", 3);
    Book algoBook("Algorithms", 1);

    dsBook.issue();
    dsBook.issue();
    algoBook.issue();
    dsBook.issue();
    dsBook.returnBook();
    dsBook.issue();
    dsBook.issue();
    dsBook.issue();

    return 0;
}
```

```
/*
Solve the problem using Object Oriented Paradigm.
A line consists of two points. The length is given by: √((x1−x2)^2 + (y1−y2)^2)
The midpoint is given by:(( x1 + x2 ) / 2 , (y1 + y2 ) / 2 )
*/
#include <iostream>
#include <cmath>

using namespace std;

class Point {
private:
    double x, y;
public:
    Point(double _x = 0, double _y = 0) : x(_x), y(_y) {}
    double getX() const { return x; }
    double getY() const { return y; }
    void setX(double _x) { x = _x; }
    void setY(double _y) { y = _y; }
};

class Line {
private:
    Point p1, p2;
public:
    Line(const Point& _p1 = Point(), const Point& _p2 = Point()) : p1(_p1), p2(_p2) {}
    double length() const {
        double dx = p1.getX() - p2.getX();
        double dy = p1.getY() - p2.getY();
        return sqrt(dx * dx + dy * dy);
    }
    Point midpoint() const {
        double x = (p1.getX() + p2.getX()) / 2;
        double y = (p1.getY() + p2.getY()) / 2;
        return Point(x, y);
    }
};

int main() {
    Point p1(0, 0);
    Point p2(3, 4);
    Line l(p1, p2);
    cout << "Length: " << l.length() << endl;
    Point mid = l.midpoint();
    cout << "Midpoint: (" << mid.getX() << ", " << mid.getY() << ")" << endl;
    return 0;
}
```

```
/*
Solve the problem using Object Oriented Paradigm.
An undergraduate course has 70 classes (14wk x 5d). A student can be present, or
absent in any of the days. Devise a way to keep track of the attendance of a
student.
- The student can be marked present or absent on day n
- The total number of present/absent has to be calculated
- Check if the student is dis-collegiate (<75%)
- Check if the student is non-collegiate (<90% & > 75%)
*/
#include <iostream>
```

```
using namespace std;
class AttendanceTracker {
private:
    int presentDays[70];
    int totalDays;
public:
    AttendanceTracker() {
        totalDays = 70;
        for (int i = 0; i < totalDays; i++) {
            presentDays[i] = 0; // 0 represents absence, 1 represents presence
        }
    }
    void markPresent(int day) {
        presentDays[day-1] = 1;
    }
    void markAbsent(int day) {
        presentDays[day-1] = 0;
    }
    int getTotalPresent() {
        int totalPresent = 0;
        for (int i = 0; i < totalDays; i++) {
            if (presentDays[i] == 1) {
                totalPresent++;
            }
        }
        return totalPresent;
    }
    int getTotalAbsent() {
        return totalDays - getTotalPresent();
    }
    bool isDisCollegiate() {
        float attendancePercentage = (getTotalPresent() / (float)totalDays) * 100;
        if (attendancePercentage < 75.0) {
            return true;
        }
        return false;
    }
    bool isNonCollegiate() {
        float attendancePercentage = (getTotalPresent() / (float)totalDays) * 100;
        if (attendancePercentage > 75.0 && attendancePercentage < 90.0) {
            return true;
        }
        return false;
    }
};

int main() {
    AttendanceTracker tracker;
    tracker.markPresent(1);
    tracker.markPresent(2);
    tracker.markAbsent(3);
    tracker.markPresent(4);
    tracker.markPresent(5);
    tracker.markPresent(6);
    tracker.markPresent(7);
    // ... mark attendance for remaining days

    cout << "Total Present: " << tracker.getTotalPresent() << endl;
    cout << "Total Absent: " << tracker.getTotalAbsent() << endl;
    cout << "Is Dis-Collegiate? " << (tracker.isDisCollegiate() ? "Yes" : "No") << endl;
    cout << "Is Non-Collegiate? " << (tracker.isNonCollegiate() ? "Yes" : "No") << endl;

    return 0;
}
```

```
/*
Problem based on namespace:
We all know about the built in function strlen(strg1). It determines the length of a
particular string strg1 (with spaces). Now you have to make your own function
int func(strg1) in a class that will find the length of a string without the
spaces.
Example:
Input: My name is x.
Built in function Output: 13
Own function Output: 10
*/
#include <iostream>
#include <string>
using namespace std;
namespace StringOperations {
    int strlenWithoutSpaces(const std::string& str) {
        int length = 0;
        for (char c : str) {
            if (c != ' ') {
                length++;
            }
        }
        return length;
    }
}

int main() {
    string str = "i'm dead inside";
    int builtInLength = str.length();
    int customLength = StringOperations::strlenWithoutSpaces(str);
    cout << "Built-in function output: " << builtInLength << endl;
    cout << "Custom function output: " << customLength << endl;
    return 0;
}
```

```
/*
Irteza likes to travel many places. Sometimes for efficient travelling he needs to
travel in minimum time or minimum cost. Naturally, he travels by BUS or TRAIN. So,
now you have to help him to find out which vehicle satisfies his requirement for
efficient travelling.
a. Write down two classes that represents a BUS and a TRAIN successively (all
the attributes will be private)
b. Each class will have two attributes
i. Velocity
ii. Cost per Kilometer
c. Write a function in each class to set the values.
d. Write one non-member function time(....) that will calculate the time
needed for each vehicle and compare them for finding the efficient vehicle.
e. Write a non-member function cost(...) that will calculate the cost for each
vehicle and compare them for finding the efficient vehicle.
*/
#include <iostream>
```

```
using namespace std;

class Bus {
private:
    double velocity; // km/h
    double costPerKm; // $
public:
    void setVelocity(double v) {
        velocity = v;
    }
    void setCostPerKm(double c) {
        costPerKm = c;
    }
    double getTime(double distance) const {
        return distance / velocity; // in hours
    }
    double getCost(double distance) const {
        return distance * costPerKm; // in dollars
    }
};

class Train {
private:
    double velocity; // km/h
    double costPerKm; // $
public:
    void setVelocity(double v) {
        velocity = v;
    }
    void setCostPerKm(double c) {
        costPerKm = c;
    }
    double getTime(double distance) const {
        return distance / velocity; // in hours
    }
    double getCost(double distance) const {
        return distance * costPerKm; // in dollars
    }
};

string time(const Bus& b, const Train& t, double distance) {
    double busTime = b.getTime(distance);
    double trainTime = t.getTime(distance);
    if (busTime < trainTime) {
        return "Bus is faster";
    } else if (trainTime < busTime) {
        return "Train is faster";
    } else {
        return "Both are equally fast";
    }
}

string cost(const Bus& b, const Train& t, double distance) {
    double busCost = b.getCost(distance);
    double trainCost = t.getCost(distance);
    if (busCost < trainCost) {
        return "Bus is cheaper";
    } else if (trainCost < busCost) {
        return "Train is cheaper";
    } else {
        return "Both cost the same";
    }
}

int main() {
    Bus b;
    b.setVelocity(80.0); // km/h
    b.setCostPerKm(0.5); // $
    Train t;
    t.setVelocity(120.0); // km/h
    t.setCostPerKm(0.7); // $
    double distance = 500.0; // km
    cout << time(b, t, distance) << endl;
    cout << cost(b, t, distance) << endl;
    return 0;
}
```

```
/*
13. In our country it is not possible to transfer money between two different banks.
You need to design a system by which transfers can be done between two different
banks(Such as DBBL and Trust Bank). For Simplicity, ignore the account number and
just think about balance. Transfer can happen in both directions, i.e., from DBBL to
Trust or from Trust to DBBL. This should be specified by user input.
a. Write down two classes that represents 2 different bank accounts such as
DBBL and Trust Bank (all the attributes will be private)
b. Each class will have 1 attribute -Balance
c. Write functions in each class to set and get the values.
d. In the main function you need to take the amount of money a user wants to
```

```
transfer.
e. Write one non-member function transferMoney(....) that will update the
Balance of each bank after each transfer.
f. Finally, show the current balance of each bank.
*/
#include <iostream>

using namespace std;

class BankAccount {
private:
    double balance;
public:
    BankAccount() {
        balance = 30000;
    }
    void setBalance(double b) {
        balance = b;
    }
    double getBalance() {
        return balance;
    }

};

void transferMoney(BankAccount& from, BankAccount& to, double amount) {
    if (from.getBalance() >= amount) {
        from.setBalance(from.getBalance() - amount);
        to.setBalance(to.getBalance() + amount);
        cout << "Transfer successful!" << endl;
    } else {
        cout << "Insufficient balance!" << endl;
    }
}

int main() {
    BankAccount dbbl;
    BankAccount trust;

    double amount;

    cout << "Enter amount to transfer: ";
    cin >> amount;

    // transfer from dbbl to trust
    transferMoney(dbbl, trust, amount);

    // transfer from trust to dbbl
    transferMoney(trust, dbbl, amount);

    // print balances
    cout << "DBBL balance: " << dbbl.getBalance() << endl;
    cout << "Trust balance: " << trust.getBalance() << endl;

    return 0;
}
```

A line can be constructed by taking multiple line segments, while each line segment consists of 2 coordinates. In this problem, you are given the x and y coordinates of n number of lines. Your first job is to construct the lines. The next job is to arrange the lines in ascending order while considering their length.

The formula for constructing a line segment, while 2 points (x1,y1) and (x2,y2) are given:

Length_of_line_segment = √((x2 – x1)2 + (y2 – y1)2)

To accomplish the jobs, you will be needed to design a class named Line, having one public method to calculate the length of the line.

The skeleton of the class is as follows:

```
class Line {
// name of the line
// no of points constructing that line (integer or float values)
// dynamically allocate memory for x and y coordinates of those points
public:
// necessary constructors, destructor
calculate_length();
};
```

Sample Input

```
Number of lines: 3
Name of Line 1: alpha
Number of points in alpha: 3
Enter coordinates:
1 1
2 2
3 4
```

```
Name of Line 2: beta
Number of points in beta: 3
Enter coordinates:
0 1
5 6
9 4
Number of points in gamma: 2
Enter coordinates:
0 0
5.2 5.6
```

Sample Output

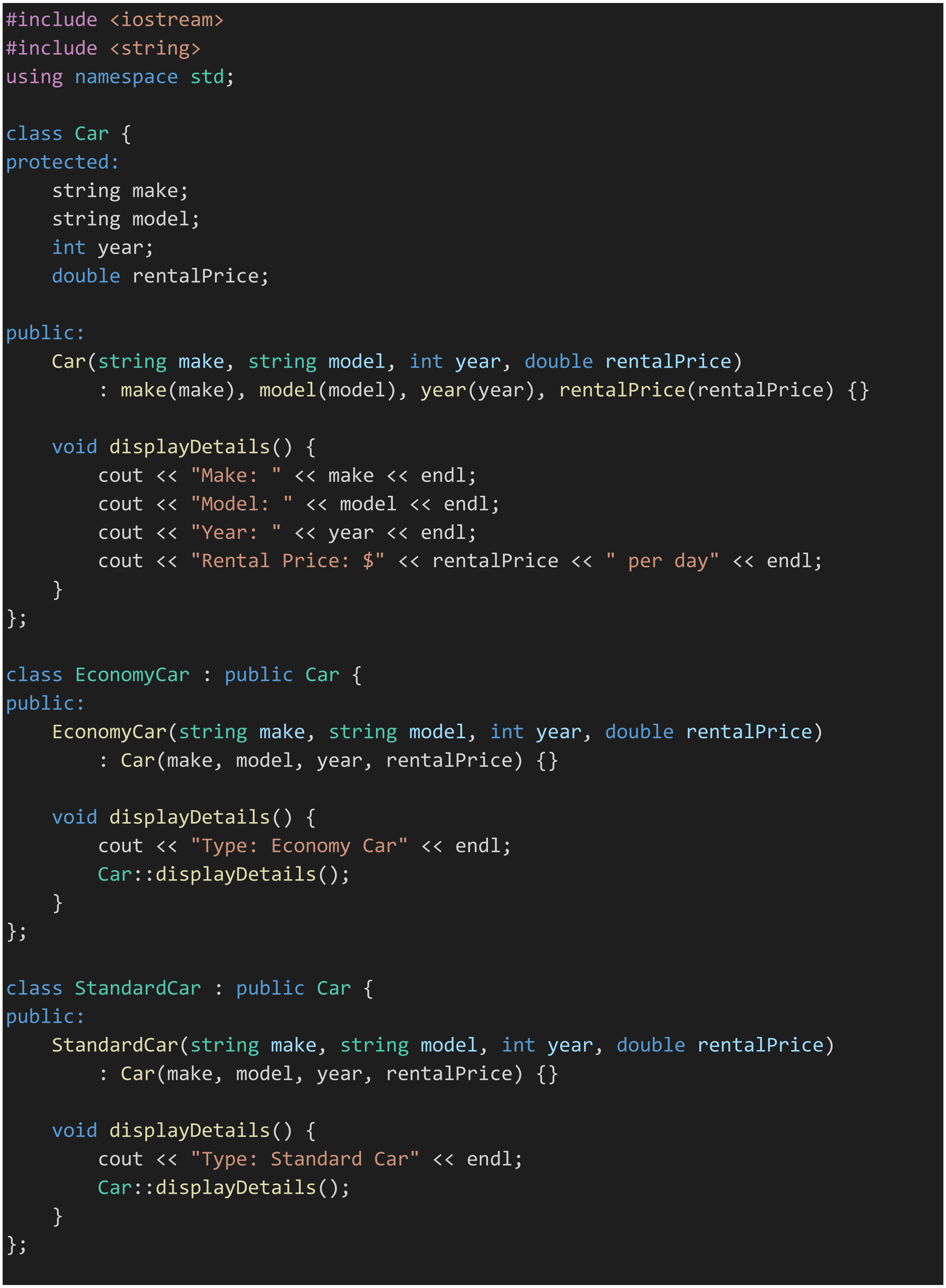
```
Length of alpha: 3.65028
Length of beta: 11.5432
Length of gamma: 7.64199
Order of Lines:
alpha<gamma<beta
```

ANSWER:

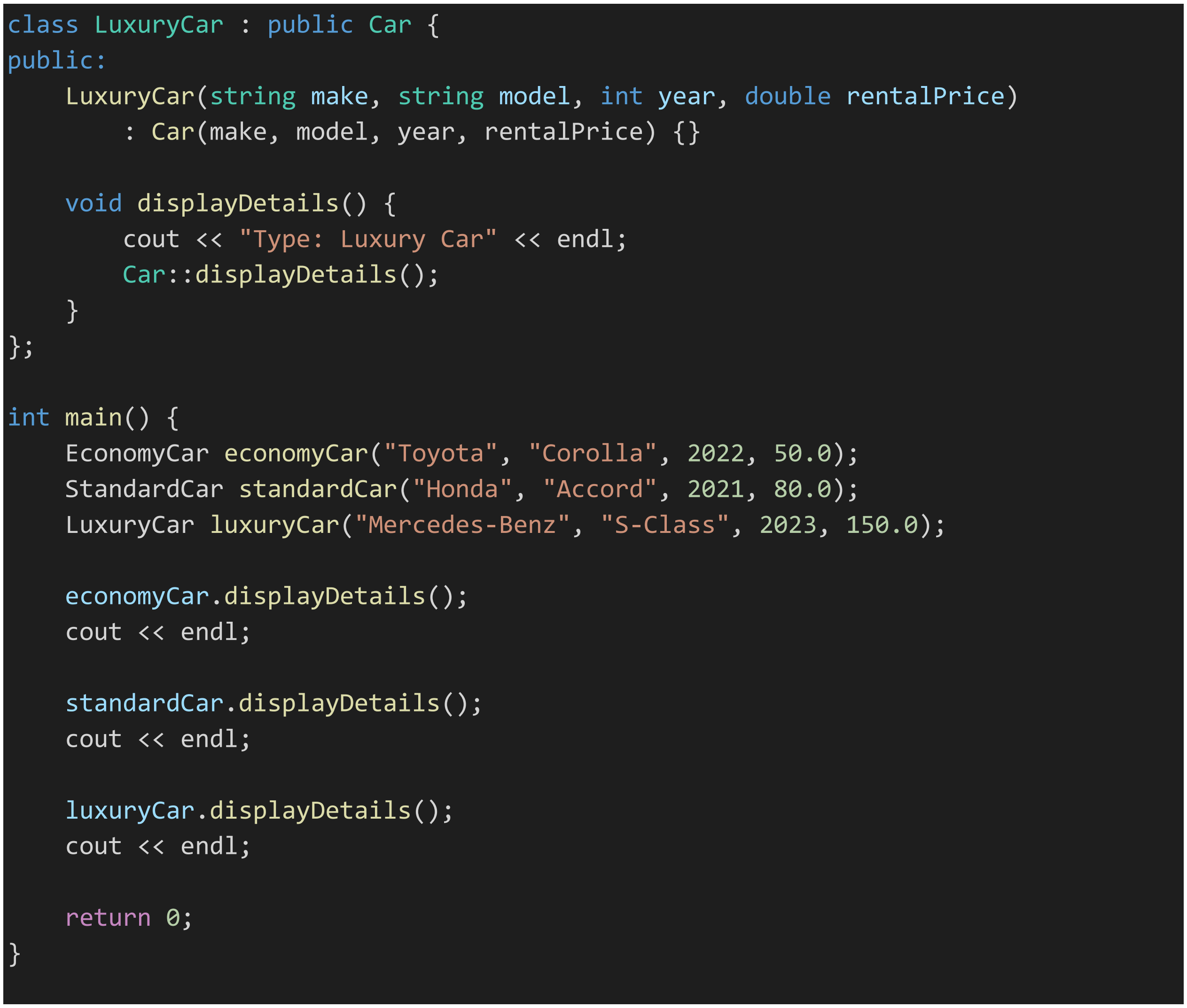
```
#include <bits/stdc++.h>
using namespace std;
class Line
{
private:
    string name;
    int length;
    float *x_coordinates;
    float *y_coordinates;
```

```
public:
    Line()
    {
        name = "";
        length = 0;
        x_coordinates = new float[length];
        y_coordinates = new float[length];
    }
    Line(string n, int l, float *x, float *y)
    {
        name = n;
        length = l;
        x_coordinates = new float[length];
        y_coordinates = new float[length];
        for (int i = 0; i < length; i++)
        {
            x_coordinates[i] = x[i];
            y_coordinates[i] = y[i];
        }
    }

    float calCulateLength()
    {
        float sum = 0;
        for (int i = 0; i < length - 1; i++)
        {
            sum += sqrt(pow(x_coordinates[i + 1] - x_coordinates[i], 2) + pow(y_coordinates[i + 1] - y_coordinates[i], 2));
        }
        return sum;
    }
    string getName()
    {
        return name;
    }
    ~Line()
    {
        delete[] x_coordinates;
        delete[] y_coordinates;
    }
};
int main()
{
    int n;
    cout << "Number of lines: ";
    cin >> n;
    Line *lines = new Line[n];
    for (int i = 0; i < n; i++)
    {
        string name;
        int length;
        cout << "Enter name of line: ";
        cin >> name;
        cout << "Enter length of line: ";
        cin >> length;
        float *x_coordinates = new float[length];
        float *y_coordinates = new float[length];
        for (int j = 0; j < length; j++)
```

```
            {
                cin >> x_coordinates[j] >> y_coordinates[j];
            }
            lines[i] = Line(name, length, x_coordinates, y_coordinates);
        }
        for (int i = 0; i < n; i++)
        {
            cout << "Length of " << lines[i].getName() << " " << lines[i].calCulateLength() << endl;
        }
        for (int i = 0; i < n; i++)
        {
            for (int j = 0; j < n - i - 1; i++)
            {
                if (lines[j].calCulateLength() > lines[j + 1].calCulateLength())
                {
                    swap(lines[j], lines[j + 1]);
                }
            }
        }
        for (int i = 0; i < n; i++)
        {
            if (i == n - 1)        {
                cout << lines[i].getName() << " ";
            }
            else{
                cout << lines[i].getName() << " < ";
            }
        }
    }
```

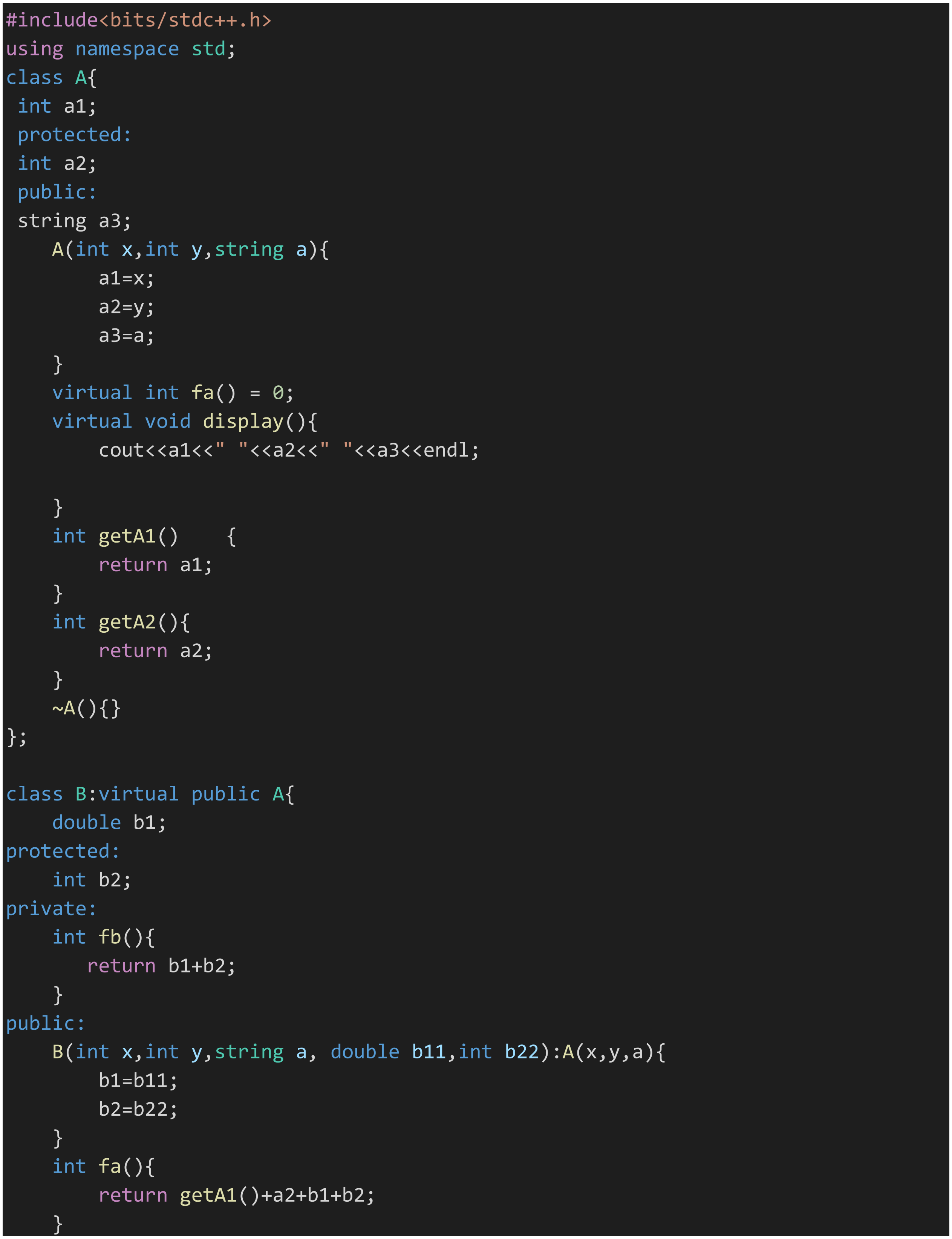


```cpp
#include <iostream>
#include <string>
using namespace std;

class Car {
protected:
    string make;
    string model;
    int year;
    double rentalPrice;

public:
    Car(string make, string model, int year, double rentalPrice)
        : make(make), model(model), year(year), rentalPrice(rentalPrice) {}

    void displayDetails() {
        cout << "Make: " << make << endl;
        cout << "Model: " << model << endl;
        cout << "Year: " << year << endl;
        cout << "Rental Price: $" << rentalPrice << " per day" << endl;
    }
};

class EconomyCar : public Car {
public:
    EconomyCar(string make, string model, int year, double rentalPrice)
        : Car(make, model, year, rentalPrice) {}

    void displayDetails() {
        cout << "Type: Economy Car" << endl;
        Car::displayDetails();
    }
};

class StandardCar : public Car {
public:
    StandardCar(string make, string model, int year, double rentalPrice)
        : Car(make, model, year, rentalPrice) {}

    void displayDetails() {
        cout << "Type: Standard Car" << endl;
        Car::displayDetails();
    }
};
```

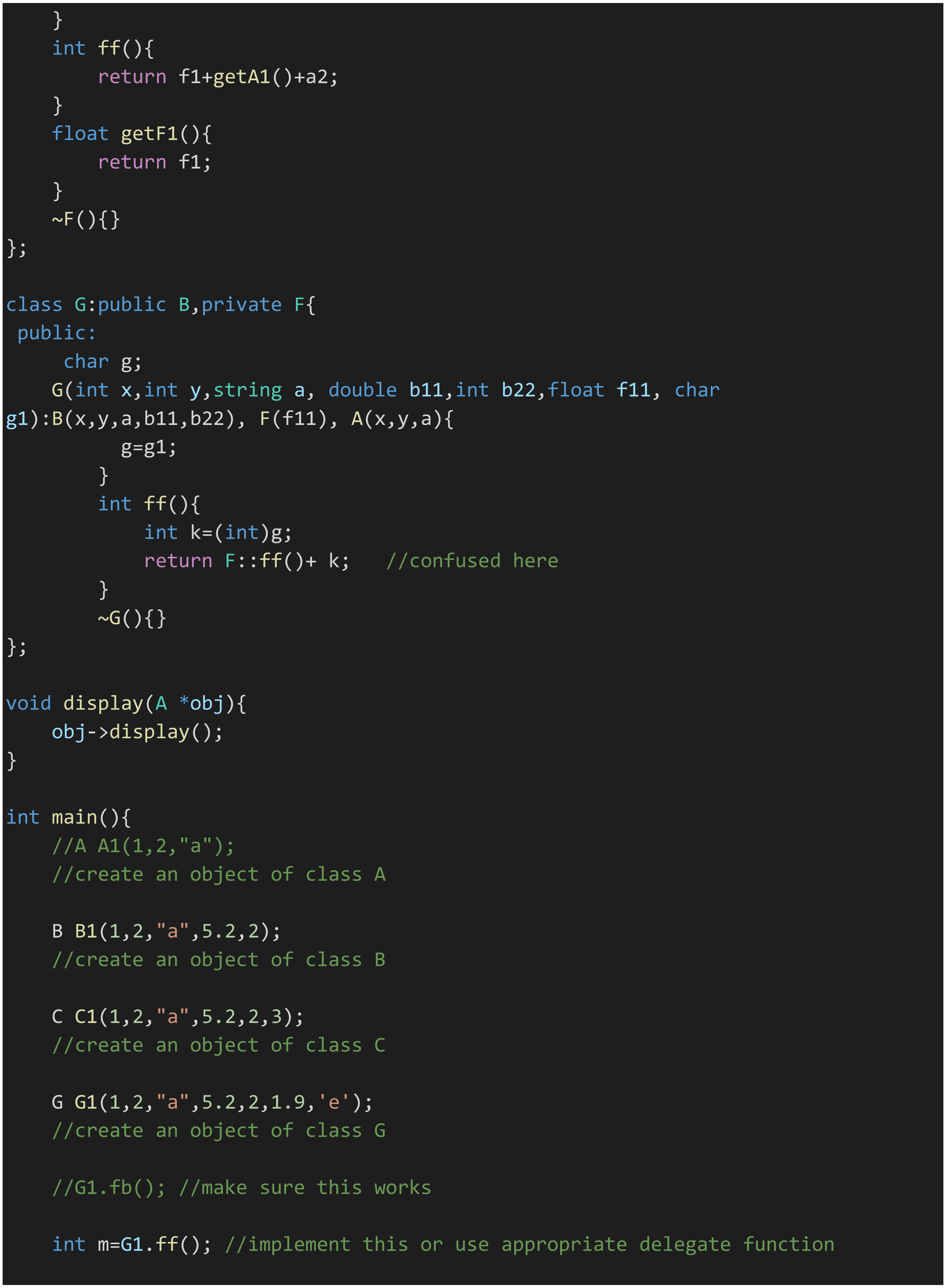


```cpp
class LuxuryCar : public Car {
public:
    LuxuryCar(string make, string model, int year, double rentalPrice)
        : Car(make, model, year, rentalPrice) {}

    void displayDetails() {
        cout << "Type: Luxury Car" << endl;
        Car::displayDetails();
    }
};

int main() {
    EconomyCar economyCar("Toyota", "Corolla", 2022, 50.0);
    StandardCar standardCar("Honda", "Accord", 2021, 80.0);
    LuxuryCar luxuryCar("Mercedes-Benz", "S-Class", 2023, 150.0);

    economyCar.displayDetails();
    cout << endl;

    standardCar.displayDetails();
    cout << endl;

    luxuryCar.displayDetails();
    cout << endl;

    return 0;
}
```

20 June MOCK Online



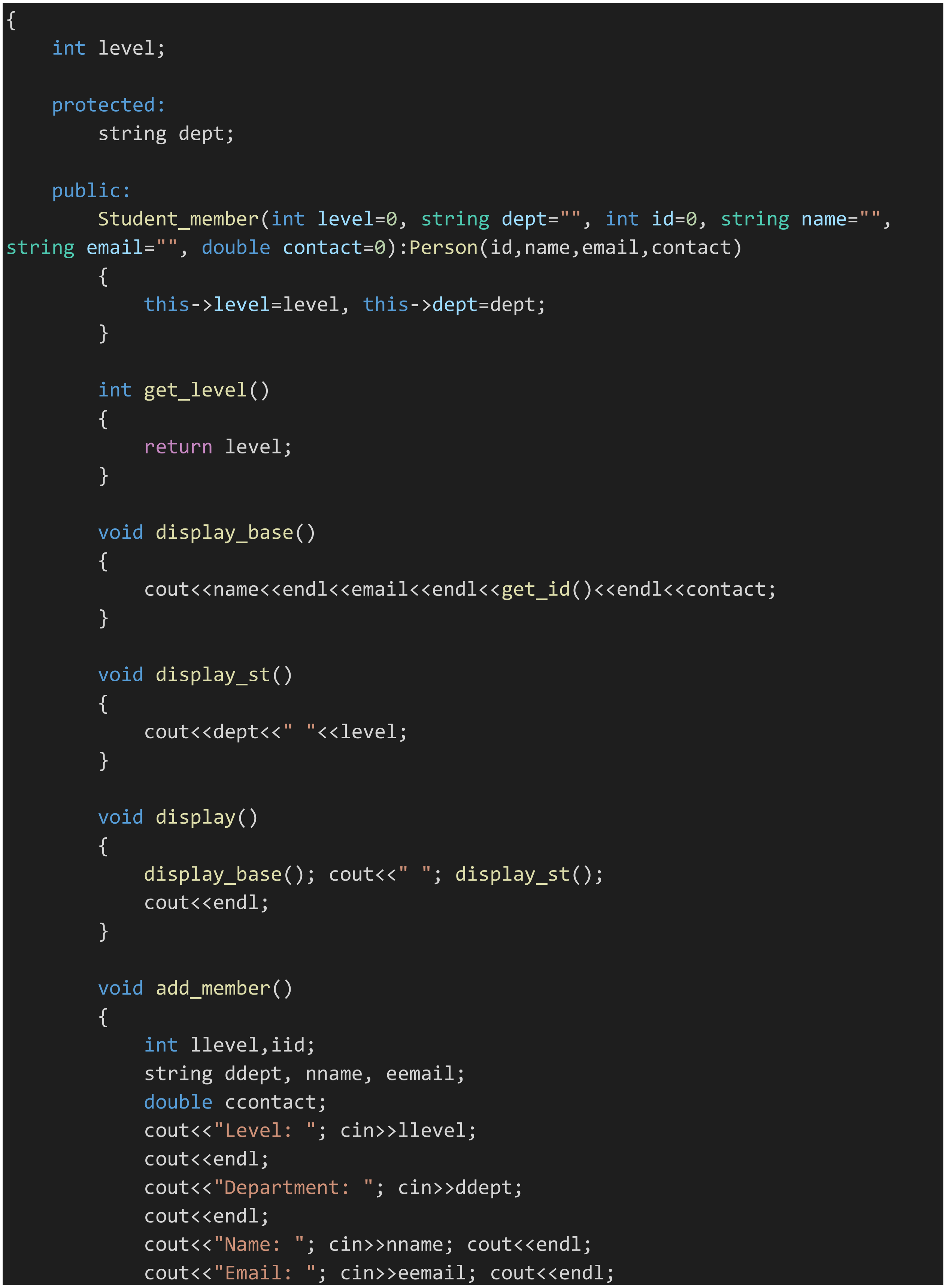
```
#include<bits/stdc++.h>
using namespace std;
class A{
 int a1;
 protected:
 int a2;
 public:
 string a3;
    A(int x,int y,string a){
        a1=x;
        a2=y;
        a3=a;
    }
    virtual int fa() = 0;
    virtual void display(){
        cout<<a1<<" "<<a2<<" "<<a3<<endl;

    }
    int getA1()     {
        return a1;
    }
    int getA2(){
        return a2;
    }
    ~A(){}
};

class B:virtual public A{
    double b1;
protected:
    int b2;
private:
    int fb(){
       return b1+b2;
    }
public:
    B(int x,int y,string a, double b11,int b22):A(x,y,a){
        b1=b11;
        b2=b22;
    }
    int fa(){
        return getA1()+a2+b1+b2;
    }
    void  display(){
        cout<<"b1 "<<b1<<" b2 "<<b2<<endl;
    }
    int call_fb(){
        return fb();
    }
    ~B(){}
};

class C:virtual public B{
public:
    int c1;
    C(int x,int y,string a, double b11,int b22,int
c11):B(x,y,a,b11,b22),A(x,y,a){
        c1=c11;
    }
    ~C(){}
};

class D:virtual public B{
 public:
     int d1;
     D(int x,int y,string a, double b11,int b22,int d11):
B(x,y,a,b11,b22),A(x,y,a)   {
         d1=d11;
     }
     ~D(){}
};

class E:public C,public D{
 public:
     int e1;
     E(int x,int y,string a, double b11,int b22,int c11,int d11,int
e11):C(x,y,a,b11,b22,c11), D(x,y,a,b11,b22,d11), B(x,y,a,b11,b22), A(x,y,a)   {
         e1=e11;
     }
     ~E(){}
};

class F: virtual public A{
 protected:
     float f1;
 public:
    F(float f11){
        f1=f11;
```



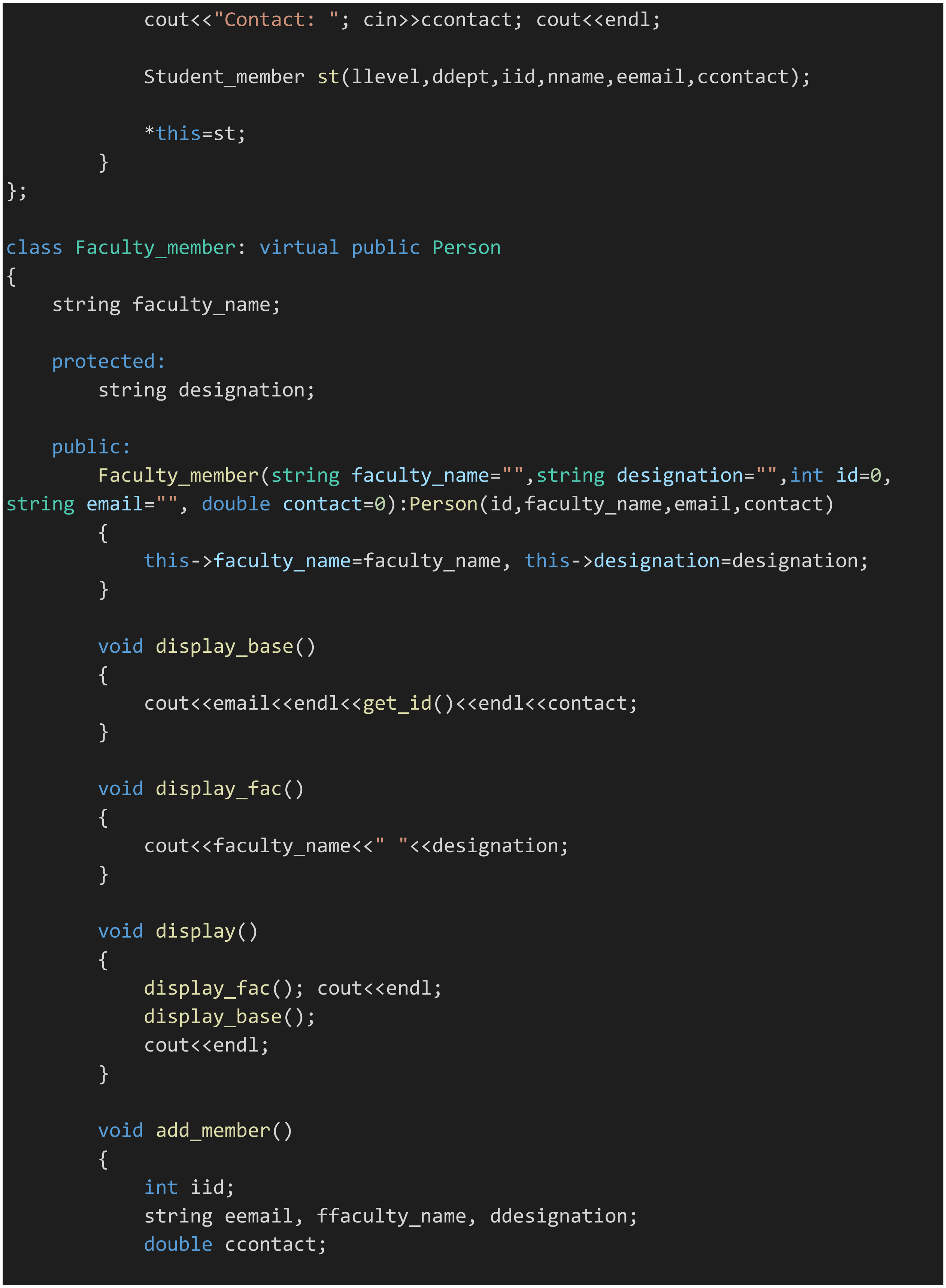
```
    }
    int ff(){
        return f1+getA1()+a2;
    }
    float getF1(){
        return f1;
    }
    ~F(){}
};

class G:public B,private F{
 public:
     char g;
    G(int x,int y,string a, double b11,int b22,float f11, char
g1):B(x,y,a,b11,b22), F(f11), A(x,y,a){
          g=g1;
        }
        int ff(){
            int k=(int)g;
            return F::ff()+ k;   //confused here
        }
        ~G(){}
};

void display(A *obj){
    obj->display();
}

int main(){
    //A A1(1,2,"a");
    //create an object of class A

    B B1(1,2,"a",5.2,2);
    //create an object of class B

    C C1(1,2,"a",5.2,2,3);
    //create an object of class C

    G G1(1,2,"a",5.2,2,1.9,'e');
    //create an object of class G

    //G1.fb(); //make sure this works

    int m=G1.ff(); //implement this or use appropriate delegate function
    cout<<m<<endl;

//Then, make sure the following lines work, use appropriate techniques as necessary
    A * ptr;
    ptr = &B1;

    ptr->display();
}
```

CSE 21 ONLINE

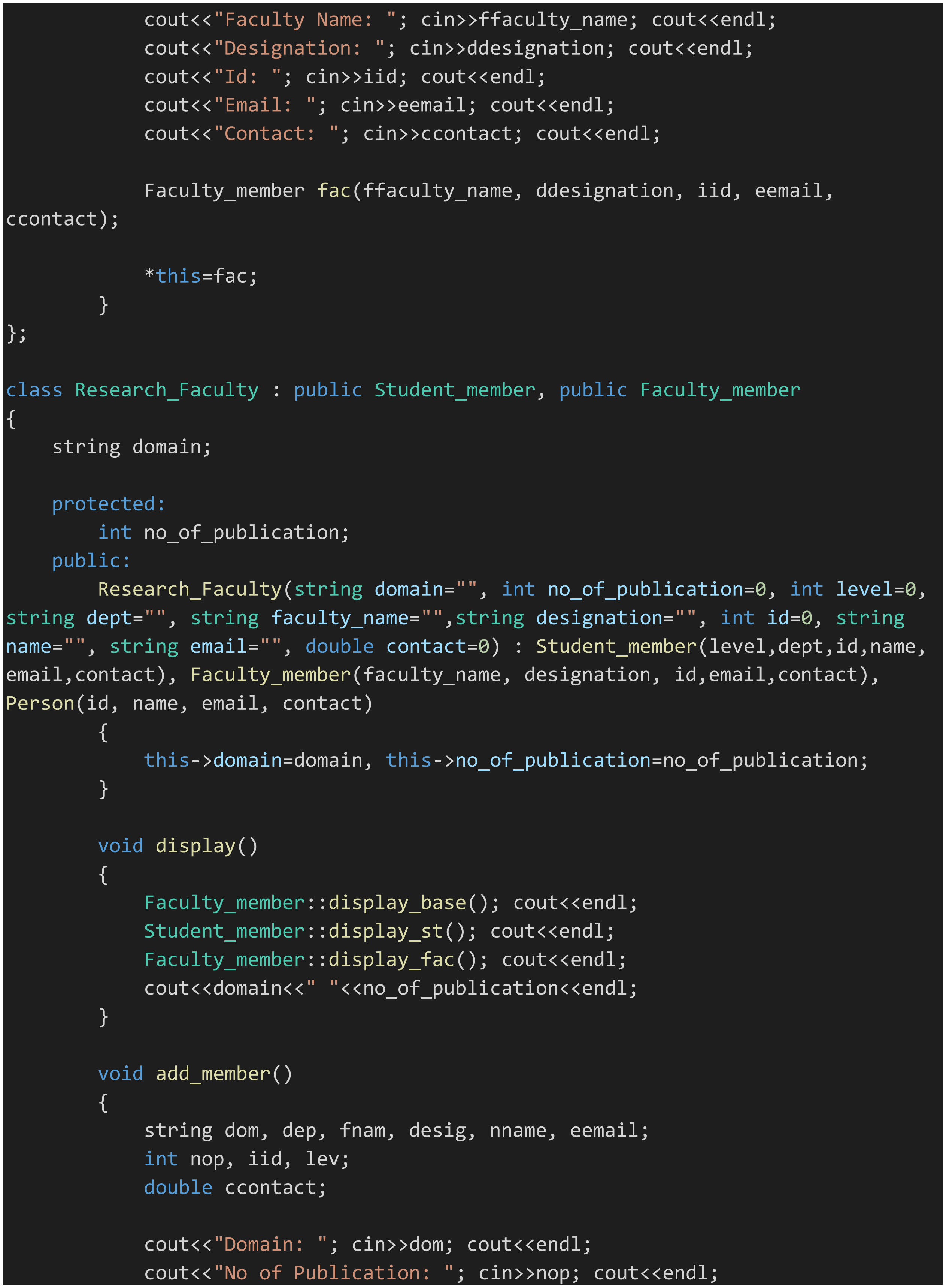
```
#include<iostream>
#include<fstream>
#include<iomanip>
#include<cmath>
#include<vector>
#include<algorithm>

using namespace std;

class Person
{
    int id;

    protected:
        string name, email;
        double contact;

    public:
        Person(int id=0, string name="", string email="", double contact=0)
        {
            this->id=id, this->name=name, this->email=email, this->contact=contact;
        }
        virtual void display()=0;
        virtual void add_member()=0;

        int get_id()
        {
            return id;
        }
};

class Student_member: virtual public Person
```



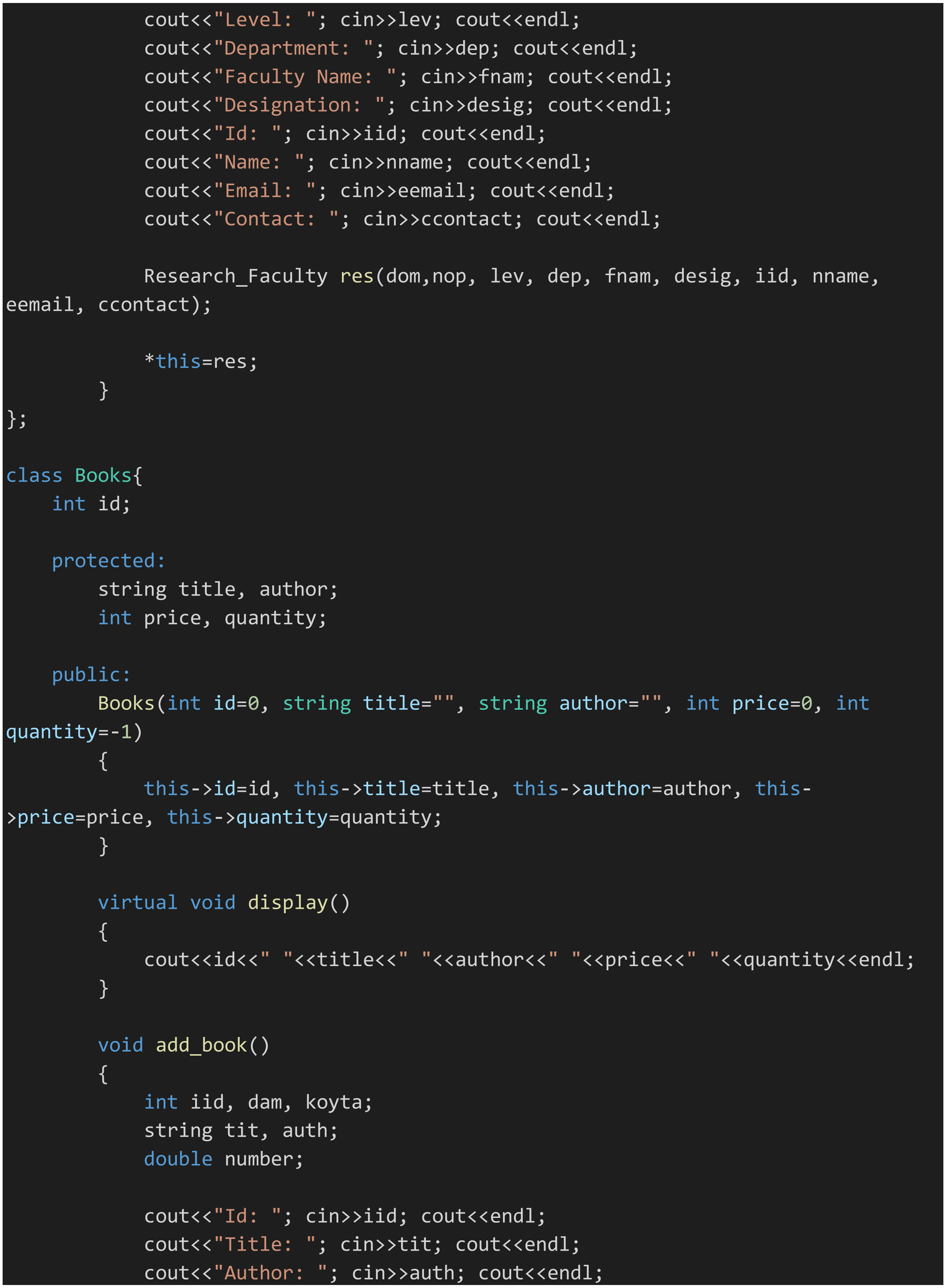
```
{
    int level;

    protected:
        string dept;

    public:
        Student_member(int level=0, string dept="", int id=0, string name="",
string email="", double contact=0):Person(id,name,email,contact)
        {
            this->level=level, this->dept=dept;
        }

        int get_level()
        {
            return level;
        }

        void display_base()
        {
            cout<<name<<endl<<email<<endl<<get_id()<<endl<<contact;
        }

        void display_st()
        {
            cout<<dept<<" "<<level;
        }

        void display()
        {
            display_base(); cout<<" "; display_st();
            cout<<endl;
        }

        void add_member()
        {
            int llevel,iid;
            string ddept, nname, eemail;
            double ccontact;
            cout<<"Level: "; cin>>llevel;
            cout<<endl;
            cout<<"Department: "; cin>>ddept;
            cout<<endl;
            cout<<"Name: "; cin>>nname; cout<<endl;
            cout<<"Email: "; cin>>eemail; cout<<endl;
```



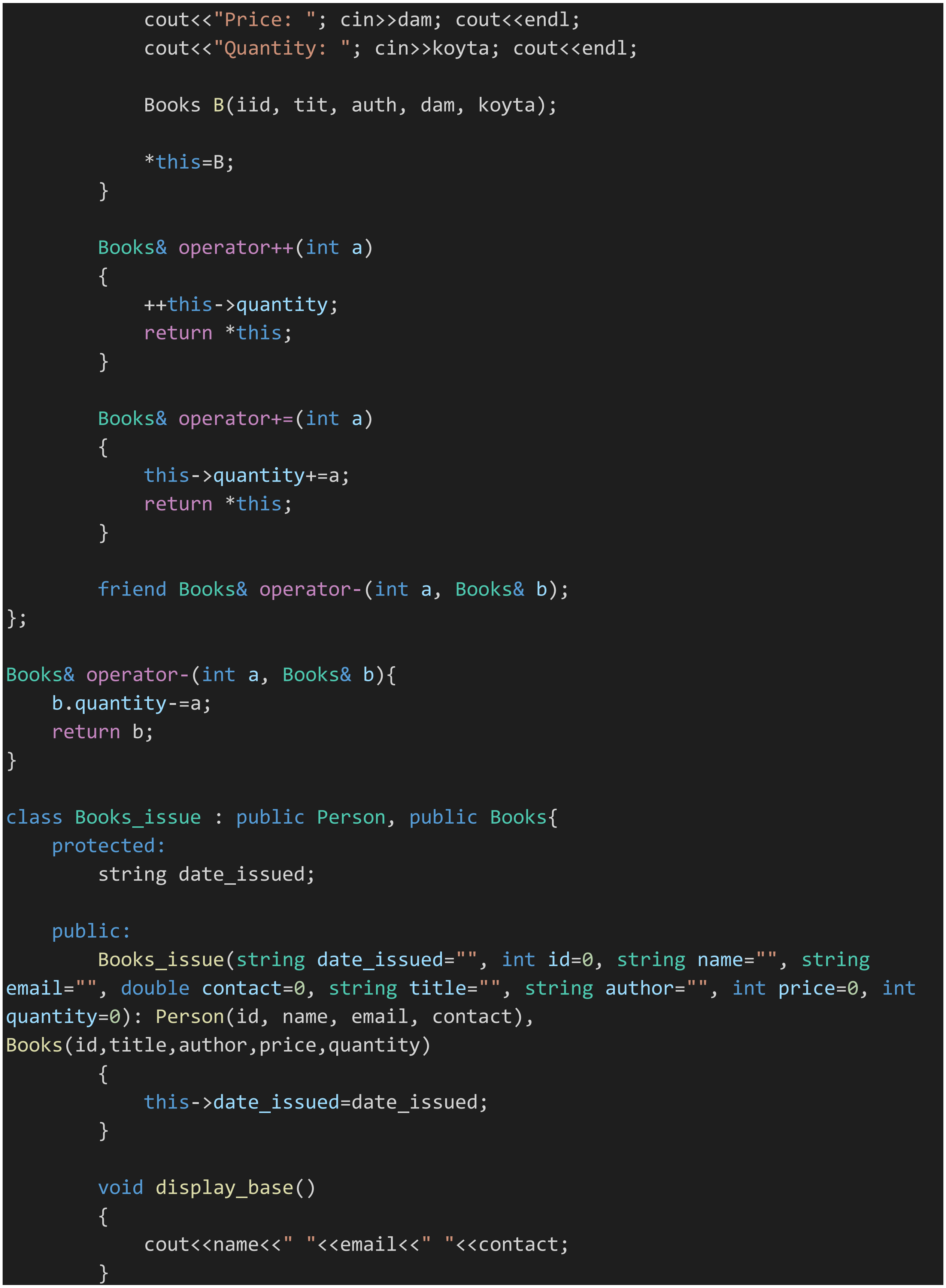
```
            cout<<"Contact: "; cin>>ccontact; cout<<endl;

            Student_member st(llevel,ddept,iid,nname,eemail,ccontact);

            *this=st;
        }
};

class Faculty_member: virtual public Person
{
    string faculty_name;

    protected:
        string designation;

    public:
        Faculty_member(string faculty_name="",string designation="",int id=0,
string email="", double contact=0):Person(id,faculty_name,email,contact)
        {
            this->faculty_name=faculty_name, this->designation=designation;
        }

        void display_base()
        {
            cout<<email<<endl<<get_id()<<endl<<contact;
        }

        void display_fac()
        {
            cout<<faculty_name<<" "<<designation;
        }

        void display()
        {
            display_fac(); cout<<endl;
            display_base();
            cout<<endl;
        }

        void add_member()
        {
            int iid;
            string eemail, ffaculty_name, ddesignation;
            double ccontact;
```



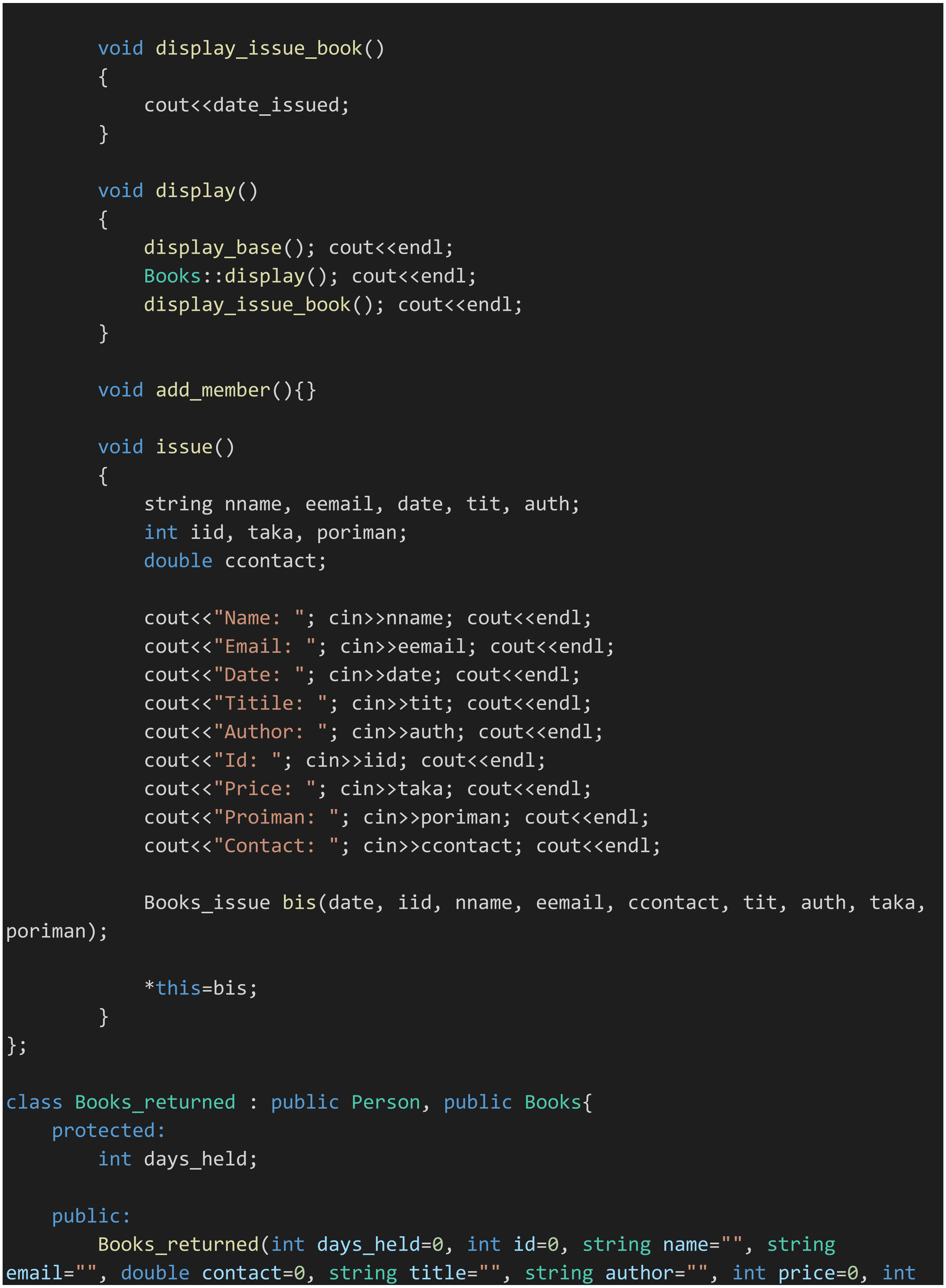
```
            cout<<"Faculty Name: "; cin>>ffaculty_name; cout<<endl;
            cout<<"Designation: "; cin>>ddesignation; cout<<endl;
            cout<<"Id: "; cin>>iid; cout<<endl;
            cout<<"Email: "; cin>>eemail; cout<<endl;
            cout<<"Contact: "; cin>>ccontact; cout<<endl;

            Faculty_member fac(ffaculty_name, ddesignation, iid, eemail,
ccontact);

            *this=fac;
        }
};

class Research_Faculty : public Student_member, public Faculty_member
{
    string domain;

    protected:
        int no_of_publication;
    public:
        Research_Faculty(string domain="", int no_of_publication=0, int level=0,
string dept="", string faculty_name="",string designation="", int id=0, string
name="", string email="", double contact=0) : Student_member(level,dept,id,name,
email,contact), Faculty_member(faculty_name, designation, id,email,contact),
Person(id, name, email, contact)
        {
            this->domain=domain, this->no_of_publication=no_of_publication;
        }

        void display()
        {
            Faculty_member::display_base(); cout<<endl;
            Student_member::display_st(); cout<<endl;
            Faculty_member::display_fac(); cout<<endl;
            cout<<domain<<" "<<no_of_publication<<endl;
        }

        void add_member()
        {
            string dom, dep, fnam, desig, nname, eemail;
            int nop, iid, lev;
            double ccontact;

            cout<<"Domain: "; cin>>dom; cout<<endl;
            cout<<"No of Publication: "; cin>>nop; cout<<endl;
```



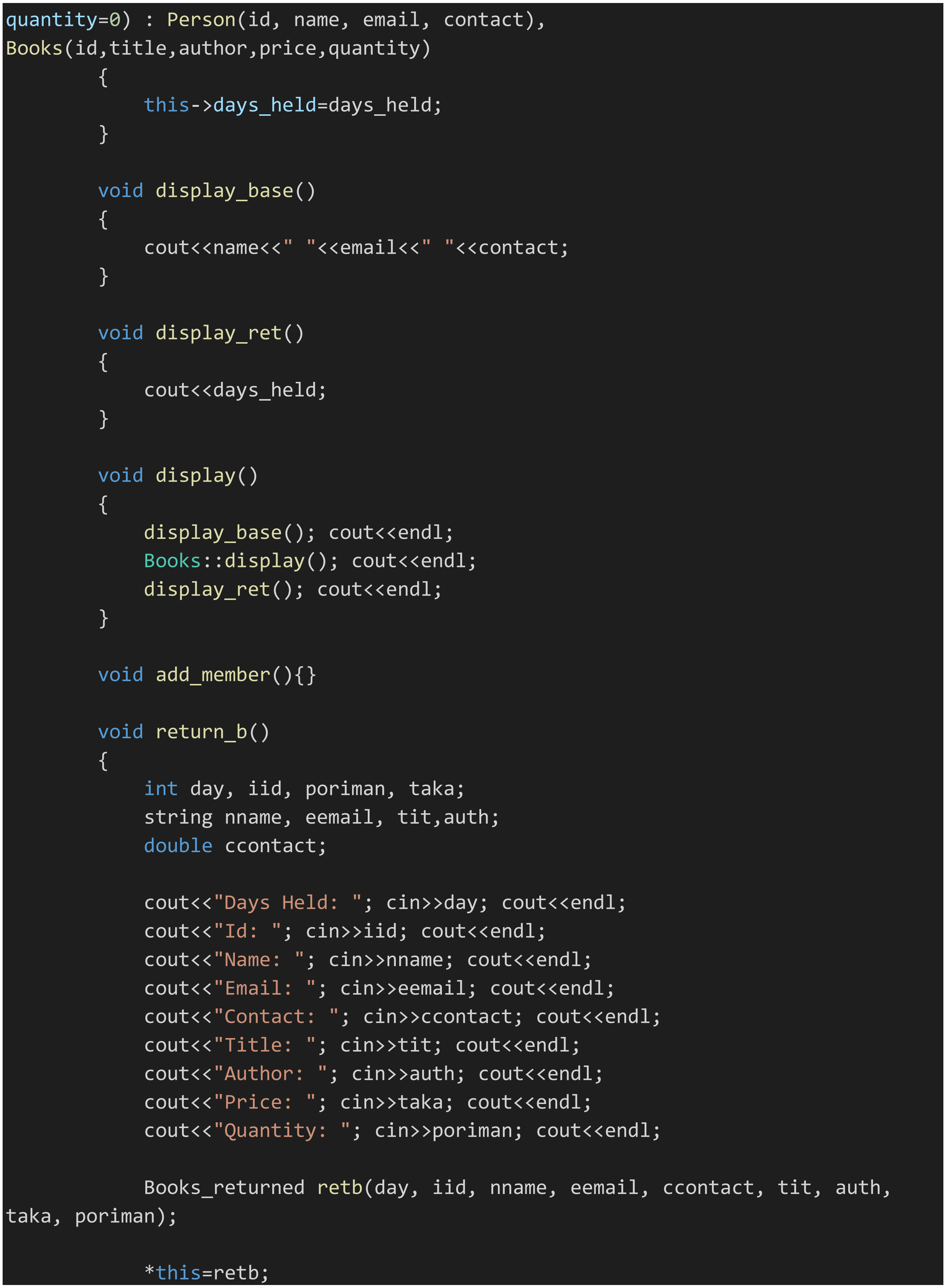
```
            cout<<"Level: "; cin>>lev; cout<<endl;
            cout<<"Department: "; cin>>dep; cout<<endl;
            cout<<"Faculty Name: "; cin>>fnam; cout<<endl;
            cout<<"Designation: "; cin>>desig; cout<<endl;
            cout<<"Id: "; cin>>iid; cout<<endl;
            cout<<"Name: "; cin>>nname; cout<<endl;
            cout<<"Email: "; cin>>eemail; cout<<endl;
            cout<<"Contact: "; cin>>ccontact; cout<<endl;

            Research_Faculty res(dom,nop, lev, dep, fnam, desig, iid, nname, eemail, ccontact);

            *this=res;
        }
};

class Books{
    int id;

    protected:
        string title, author;
        int price, quantity;

    public:
        Books(int id=0, string title="", string author="", int price=0, int quantity=-1)
        {
            this->id=id, this->title=title, this->author=author, this->price=price, this->quantity=quantity;
        }

        virtual void display()
        {
            cout<<id<<" "<<title<<" "<<author<<" "<<price<<" "<<quantity<<endl;
        }

        void add_book()
        {
            int iid, dam, koyta;
            string tit, auth;
            double number;

            cout<<"Id: "; cin>>iid; cout<<endl;
            cout<<"Title: "; cin>>tit; cout<<endl;
            cout<<"Author: "; cin>>auth; cout<<endl;
```



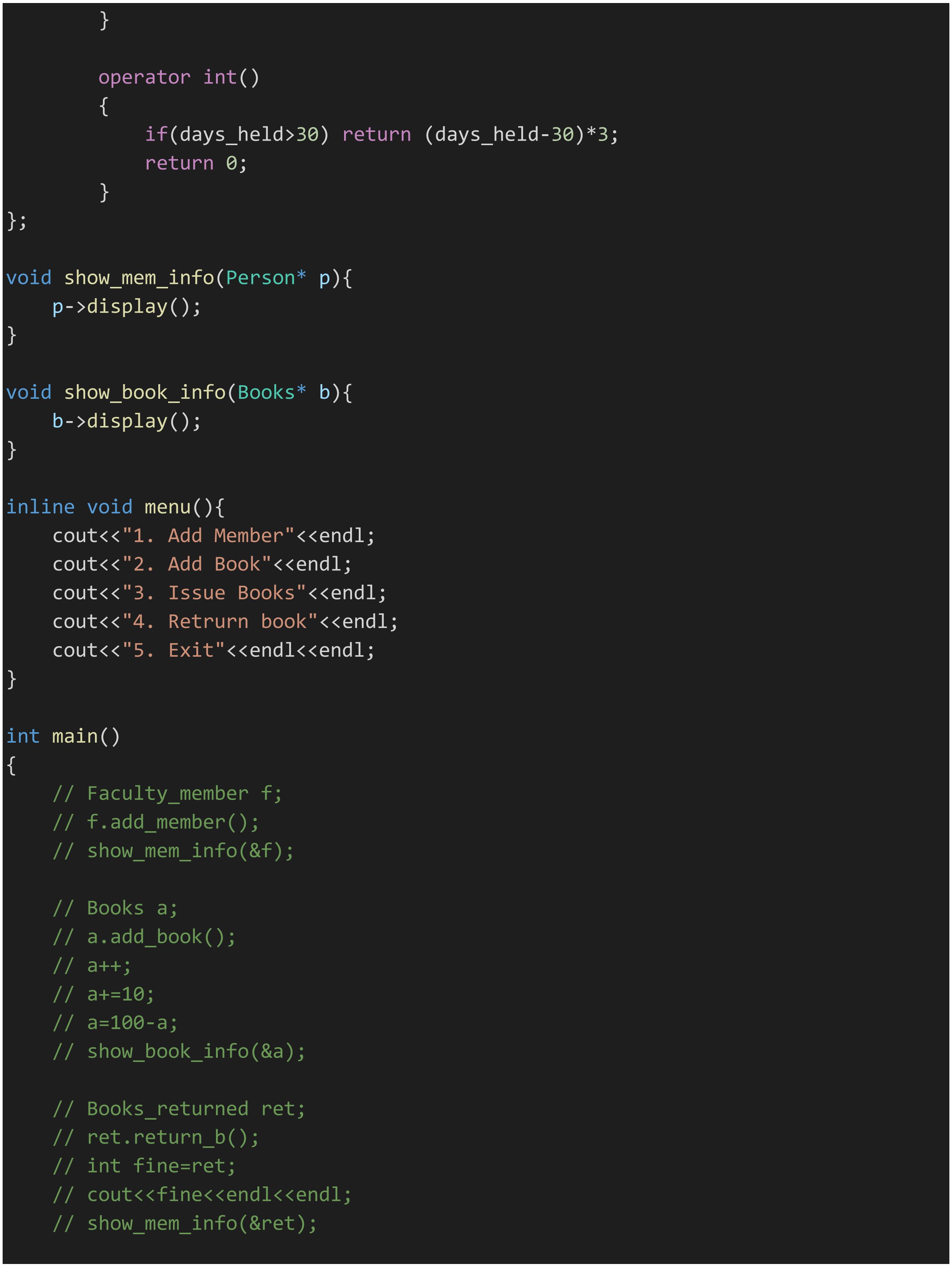
```
            cout<<"Price: "; cin>>dam; cout<<endl;
            cout<<"Quantity: "; cin>>koyta; cout<<endl;

            Books B(iid, tit, auth, dam, koyta);

            *this=B;
        }

        Books& operator++(int a)
        {
            ++this->quantity;
            return *this;
        }

        Books& operator+=(int a)
        {
            this->quantity+=a;
            return *this;
        }

        friend Books& operator-(int a, Books& b);
};

Books& operator-(int a, Books& b){
    b.quantity-=a;
    return b;
}

class Books_issue : public Person, public Books{
    protected:
        string date_issued;

    public:
        Books_issue(string date_issued="", int id=0, string name="", string
email="", double contact=0, string title="", string author="", int price=0, int
quantity=0): Person(id, name, email, contact),
Books(id,title,author,price,quantity)
        {
            this->date_issued=date_issued;
        }

        void display_base()
        {
            cout<<name<<" "<<email<<" "<<contact;
        }
```



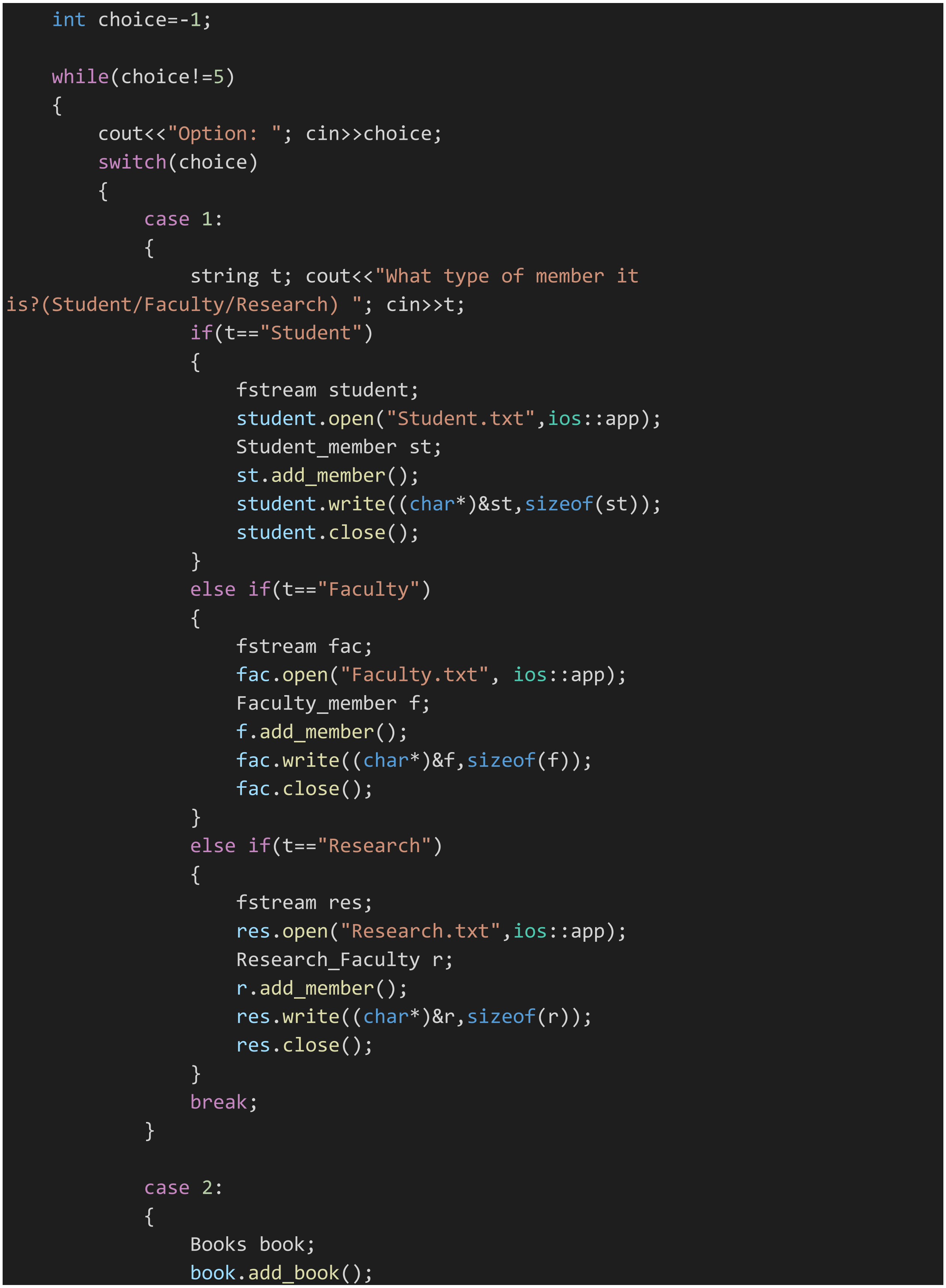
```
        void display_issue_book()
        {
            cout<<date_issued;
        }

        void display()
        {
            display_base(); cout<<endl;
            Books::display(); cout<<endl;
            display_issue_book(); cout<<endl;
        }

        void add_member(){}

        void issue()
        {
            string nname, eemail, date, tit, auth;
            int iid, taka, poriman;
            double ccontact;

            cout<<"Name: "; cin>>nname; cout<<endl;
            cout<<"Email: "; cin>>eemail; cout<<endl;
            cout<<"Date: "; cin>>date; cout<<endl;
            cout<<"Titile: "; cin>>tit; cout<<endl;
            cout<<"Author: "; cin>>auth; cout<<endl;
            cout<<"Id: "; cin>>iid; cout<<endl;
            cout<<"Price: "; cin>>taka; cout<<endl;
            cout<<"Proiman: "; cin>>poriman; cout<<endl;
            cout<<"Contact: "; cin>>ccontact; cout<<endl;

            Books_issue bis(date, iid, nname, eemail, ccontact, tit, auth, taka,
poriman);

            *this=bis;
        }
};

class Books_returned : public Person, public Books{
    protected:
        int days_held;

    public:
        Books_returned(int days_held=0, int id=0, string name="", string
email="", double contact=0, string title="", string author="", int price=0, int
```



```
quantity=0) : Person(id, name, email, contact),
Books(id,title,author,price,quantity)
        {
            this->days_held=days_held;
        }

        void display_base()
        {
            cout<<name<<" "<<email<<" "<<contact;
        }

        void display_ret()
        {
            cout<<days_held;
        }

        void display()
        {
            display_base(); cout<<endl;
            Books::display(); cout<<endl;
            display_ret(); cout<<endl;
        }

        void add_member(){}

        void return_b()
        {
            int day, iid, poriman, taka;
            string nname, eemail, tit,auth;
            double ccontact;

            cout<<"Days Held: "; cin>>day; cout<<endl;
            cout<<"Id: "; cin>>iid; cout<<endl;
            cout<<"Name: "; cin>>nname; cout<<endl;
            cout<<"Email: "; cin>>eemail; cout<<endl;
            cout<<"Contact: "; cin>>ccontact; cout<<endl;
            cout<<"Title: "; cin>>tit; cout<<endl;
            cout<<"Author: "; cin>>auth; cout<<endl;
            cout<<"Price: "; cin>>taka; cout<<endl;
            cout<<"Quantity: "; cin>>poriman; cout<<endl;

            Books_returned retb(day, iid, nname, eemail, ccontact, tit, auth,
taka, poriman);

            *this=retb;
```



```
        }

        operator int()
        {
            if(days_held>30) return (days_held-30)*3;
            return 0;
        }
};

void show_mem_info(Person* p){
    p->display();
}

void show_book_info(Books* b){
    b->display();
}

inline void menu(){
    cout<<"1. Add Member"<<endl;
    cout<<"2. Add Book"<<endl;
    cout<<"3. Issue Books"<<endl;
    cout<<"4. Retrurn book"<<endl;
    cout<<"5. Exit"<<endl<<endl;
}

int main()
{
    // Faculty_member f;
    // f.add_member();
    // show_mem_info(&f);

    // Books a;
    // a.add_book();
    // a++;
    // a+=10;
    // a=100-a;
    // show_book_info(&a);

    // Books_returned ret;
    // ret.return_b();
    // int fine=ret;
    // cout<<fine<<endl<<endl;
    // show_mem_info(&ret);
```



```
    int choice=-1;

    while(choice!=5)
    {
        cout<<"Option: "; cin>>choice;
        switch(choice)
        {
            case 1:
            {
                string t; cout<<"What type of member it
is?(Student/Faculty/Research) "; cin>>t;
                if(t=="Student")
                {
                    fstream student;
                    student.open("Student.txt",ios::app);
                    Student_member st;
                    st.add_member();
                    student.write((char*)&st,sizeof(st));
                    student.close();
                }
                else if(t=="Faculty")
                {
                    fstream fac;
                    fac.open("Faculty.txt", ios::app);
                    Faculty_member f;
                    f.add_member();
                    fac.write((char*)&f,sizeof(f));
                    fac.close();
                }
                else if(t=="Research")
                {
                    fstream res;
                    res.open("Research.txt",ios::app);
                    Research_Faculty r;
                    r.add_member();
                    res.write((char*)&r,sizeof(r));
                    res.close();
                }
                break;
            }

            case 2:
            {
                Books book;
                book.add_book();
                fstream b;
                b.open("book.log",ios::app);
                b.write((char*)&book,sizeof(book));
                b.close();
                break;
            }

            case 3:
            {
                Books_issue B;
                B.issue();
                break;
            }

            case 4:
            {
                Books_returned B;
                B.return_b();
                break;
            }
        }
    }

    return 0;
}

/*Test Case 1: Adding a Student Member

1. Add Member
What type of member it is? (Student/Faculty/Research) Student
Level: 2
Department: Computer Science
Name: John Doe
Email: john@example.com
Contact: 1234567890
Test Case 2: Adding a Faculty Member


1. Add Member
What type of member it is? (Student/Faculty/Research) Faculty
Faculty Name: Jane Smith
Designation: Professor
Id: 1001
Email: jane@example.com
Contact: 9876543210
```

```
Test Case 3: Adding a Research Faculty Member

1. Add Member
What type of member it is? (Student/Faculty/Research) Research
Domain: Artificial Intelligence
No of Publication: 5
Level: 3
Department: Computer Science
Faculty Name: John Smith
Designation: Assistant Professor
Id: 1002
Name: Alice Johnson
Email: alice@example.com
Contact: 8765432109
Test Case 4: Adding a Book


2. Add Book
Id: 1
Title: Introduction to Programming
Author: John Smith
Price: 50
Quantity: 10
Test Case 5: Issuing a Book


3. Issue Books
Name: John Doe
Email: john@example.com
Date: 2023-05-11
Titile: Introduction to Programming
Author: John Smith
Id: 1
Price: 50
Proiman: 1
Contact: 1234567890
Test Case 6: Returning a Book

4. Return book
Days Held: 25
Id: 1
Name: John Doe
Email: john@example.com
Contact: 1234567890
Title: Introduction to Programming
```

```
Author: John Smith
Price: 50
Quantity: 1*/
```

Define a namespace named "Inventory".Under the “Inventory” namespace define another

namespace named “Electronics”.Under the “Electronics” namespace define a class named

“Smartphone”.

- Private attributes of the “Smartphone” class are model name, price, quantity

- Take an array (size=10) of object for the “Smartphone” class and then take phone model

name, price and quantity as user input and set them through the necessary function of the

class.

- Write a menu-driven program where the admin can add phone models and their

corresponding attributes(price and quantity) and display them

Sample Input-Output

```
Welcome to the Inventory

1. Add phone model on inventory

2. Display phone inventory

Enter Option: 1

Enter the model name: Samsung

Galaxy A14

Enter the price of the model: 21,999

Enter the quantity of model: 80

1. Add phone model on inventory

2. Display phone inventory

Enter Option: 1

Enter the model name: Realme C55
Enter the price of the model: 22,999
Enter the quantity of the model: 100
1. Add phone model on inventory
2. Display phone inventory
Enter Option: 1
Enter the model name: Infinix Hot 30
OUTPUT:
Enter the price of the model: 17,999
Enter the quantity of model: 40
1. Add phone model on inventory
2. Display phone inventory
Enter Option: 2
Model Name Price Quantity
Samsung Galaxy A14 21,999 80
Realme C55 22,999 100
Infinix Hot 30 17,999 40
```

Answer:

```
#include <iostream>
#include <string>

namespace Inventory {
    namespace Electronics {
        const int MAX_SMARTPHONES = 10;

        class Smartphone {
        private:
            std::string model_name;
            double price;
            int quantity;
        public:
            Smartphone() : model_name(""), price(0), quantity(0) {}

            void setModelName(const std::string& name) {
                model_name = name;
            }

            void setPrice(double price) {
                this->price = price;
            }

            void setQuantity(int quantity) {
                this->quantity = quantity;
            }

            std::string getModelName() const {
                return model_name;
            }

            double getPrice() const {
                return price;
            }

            int getQuantity() const {
                return quantity;
            }
        };

        Smartphone smartphoneInventory[MAX_SMARTPHONES];
```

```
        int totalSmartphones = 0;
    }
}

void addPhoneModel() {
    using namespace Inventory::Electronics;

    if (totalSmartphones >= Smartphone::MAX_SMARTPHONES) {
        std::cout << "Inventory is full. Cannot add more phone models." << std::endl;
        return;
    }

    std::string modelName;
    double price;
    int quantity;

    std::cout << "Enter the model name: ";
    std::cin.ignore();
    std::getline(std::cin, modelName);

    std::cout << "Enter the price of the model: ";
    std::cin >> price;

    std::cout << "Enter the quantity of the model: ";
    std::cin >> quantity;

    Smartphone phone;
    phone.setModelName(modelName);
    phone.setPrice(price);
    phone.setQuantity(quantity);

    smartphoneInventory[totalSmartphones++] = phone;

    std::cout << "Phone model added to the inventory." << std::endl;
}

void displayPhoneInventory() {
    using namespace Inventory::Electronics;

    if (totalSmartphones == 0) {
        std::cout << "Inventory is empty. No phone models to display." << std::endl;
        return;
    }

    std::cout << "Model Name\tPrice\tQuantity" << std::endl;
    for (int i = 0; i < totalSmartphones; i++) {
        const Smartphone& phone = smartphoneInventory[i];
        std::cout << phone.getModelName() << "\t" << phone.getPrice() << "\t" << phone.getQuantity() << std::endl;
    }
}

int main() {
    using namespace std;

    cout << "Welcome to the Inventory" << endl;

    int option;
    do {
        cout << "1. Add phone model to inventory" << endl;
        cout << "2. Display phone inventory" << endl;
        cout << "Enter Option: ";
        cin >> option;

        switch (option) {
            case 1:
                addPhoneModel();
                break;
            case 2:
                displayPhoneInventory();
                break;
            default:
                cout << "Invalid option. Please try again." << endl;
        }

    } while (option == 1 || option == 2);

    return 0;
}
```

You have been assigned to develop a software application for a music streaming service. The application requires to store info about songs and playlists. Each song has attributes such as title, artist, and duration, number of views, while each playlist has attributes such as a name and a dynamic list, to contain the songs in the playlist. Additionally, the classes should include member functions to add songs to a playlist, delete a song from the playlist and display the songs in a playlist.

Now, based on the scenario given above, design the required number of classes, with the appropriate constructos, getter-setter methods, member functions and destructors, to fulfill the demand of the software application. Moreover, formulate a non-member function, which will identify the most popular song of the playlist. The song having the most number of views is the most popular song. Note that the size of the playlist for a particular user is fixed (n=5).

Interface of the Software Application

```
Application Options:
1. Create a playlist
2. Diplay the playlists
Enter Option: 1
Playlist Name: MyFavourites
MyFavourites OPTIONS:
1. Add a song
2. Delete a song
3. Display the songs
4. Most popular song
5. Return to main menu
Enter Playlist Option: 1
Info about a song:
Title: ABC
Artist: Fahim Shikdar
Duration: 5 min 20s
Views: 200
ADDED Successfully :’)
```

MyFavourites OPTIONS:

1. Add a song

2. Delete a song

3. Display the songs

4. Most popular song

5. Return to main menu

Enter Playlist Option: 1

Info about a song:

Title: DEF

Artist: Mehedi Moon

Duration: 3 min 45s

Views: 350

Generate the code in C++

ADDED Successfully :’)

MyFavourites OPTIONS:

1. Add a song

2. Delete a song

3. Display the songs

4. Most popular song

5. Return to main menu

Enter Option: 3

Songs: ABC, DEF

MyFavourites OPTIONS:

1. Add a song

2. Delete a song

3. Display the songs

4. Most popular song

5. Return to main menu

Enter Playlist Option: 4

```
Most Popular Song: DEF
MyFavourites OPTIONS:
1. Add a song
2. Delete a song
3. Display the songs
4. Most popular song
5. Return to main menu
Enter Playlist Option: 2
Song Name: DEF
MyFavourites OPTIONS:
1. Add a song
2. Delete a song
3. Display the songs
4. Most popular song
5. Return to main menu
Enter Playlist Option: 3
Songs: ABC
MyFavourites OPTIONS:
1. Add a song
2. Delete a song
3. Display the songs
4. Most popular song
5. Return to main menu
Enter Playlist Option: 5
Returning to the main menu...
Application Options:
1. Create a playlist
2. Diplay the playlists
Enter Option: 1
```

```
Playlist Name: BestOf2023

BestOf2023 OPTIONS:

1. Add a song

2. Delete a song

3. Display the songs

4. Most popular song

5. Return to main menu

Enter Playlist Option: 5

Returning to the main

menu...

Application Options:

1. Create a playlist

2. Diplay the playlists

Enter Option: 2

Playlists: MyFavourites,

BestOf2023
```

```
#include <iostream>
#include <vector>
#include <string>
#include <algorithm>

class Song {
private:
    std::string title;
    std::string artist;
    std::string duration;
    int views;

public:
    Song(const std::string& title, const std::string& artist, const std::string& duration, int views)
        : title(title), artist(artist), duration(duration), views(views) {}

    std::string getTitle() const {
        return title;
    }

    int getViews() const {
        return views;
    }
};

class Playlist {
private:
    std::string name;
    std::vector<Song> songs;
```

```
public:
    Playlist(const std::string& name) : name(name) {}

    void addSong(const Song& song) {
        songs.push_back(song);
    }

    void deleteSong(const std::string& title) {
        songs.erase(std::remove_if(songs.begin(), songs.end(),
            [&](const Song& song) { return song.getTitle() == title; }), songs.end());
    }

    void displaySongs() const {
        if (songs.empty()) {
            std::cout << "No songs in the playlist." << std::endl;
            return;
        }

        std::cout << "Songs: ";
        for (const Song& song : songs) {
            std::cout << song.getTitle() << ", ";
        }
        std::cout << std::endl;
    }

    std::string getMostPopularSong() const {
        if (songs.empty()) {
            return "No songs in the playlist.";
        }

        const Song& mostPopular = *std::max_element(songs.begin(), songs.end(),
            [](const Song& song1, const Song& song2) { return song1.getViews() < song2.getViews(); });

        return mostPopular.getTitle();
    }
};

int main() {
    std::vector<Playlist> playlists;

    while (true) {
        std::cout << "Application Options:" << std::endl;
        std::cout << "1. Create a playlist" << std::endl;
        std::cout << "2. Display the playlists" << std::endl;
        std::cout << "Enter Option: ";

        int option;
        std::cin >> option;

        if (option == 1) {
            std::string playlistName;
            std::cout << "Playlist Name: ";
            std::cin.ignore();
            std::getline(std::cin, playlistName);

            Playlist playlist(playlistName);
            playlists.push_back(playlist);
```

```
while (true) {
  std::cout << playlistName << " OPTIONS:" << std::endl;
  std::cout << "1. Add a song" << std::endl;
  std::cout << "2. Delete a song" << std::endl;
  std::cout << "3. Display the songs" << std::endl;
  std::cout << "4. Most popular song" << std::endl;
  std::cout << "5. Return to main menu" << std::endl;
  std::cout << "Enter Playlist Option: ";

  int playlistOption;
  std::cin >> playlistOption;

  if (playlistOption == 1) {
    std::string title, artist, duration;
    int views;

    std::cout << "Info about a song:" << std::endl;
    std::cout << "Title: ";
    std::cin.ignore();
    std::getline(std::cin, title);

    std::cout << "Artist: ";
    std::getline(std::cin, artist);

    std::cout << "Duration: ";
    std::getline(std::cin, duration);

    std::cout << "Views: ";
```

```
            std::cin >> views;

            Song song(title, artist, duration, views);
            playlist.addSong(song);

            std::cout << "ADDED Successfully :’)" << std::endl;
        }
        else if (playlistOption == 2) {
            std::string title;
            std::cout << "Song Name: ";
            std::cin.ignore();
            std::getline(std::cin, title);

            playlist.deleteSong(title);
        }
        else if (playlistOption == 3) {
            playlist.displaySongs();
        }
        else if (playlistOption == 4) {
            std::cout << "Most Popular Song: " << playlist.getMostPopularSong() << std::endl;
        }
        else if (playlistOption == 5) {
            std::cout << "Returning to the main menu..." << std::endl;
            break;
        }
        else {
            std::cout << "Invalid option. Please try again." << std::endl;
        }
    }
```

```
        }
        else if (option == 2) {
            std::cout << "Playlists: ";
            for (const Playlist& playlist : playlists) {
                std::cout << playlistName << ", ";
            }
            std::cout << std::endl;
        }
        else {
            std::cout << "Invalid option. Please try again." << std::endl;
        }
    }

    return 0;
}
```