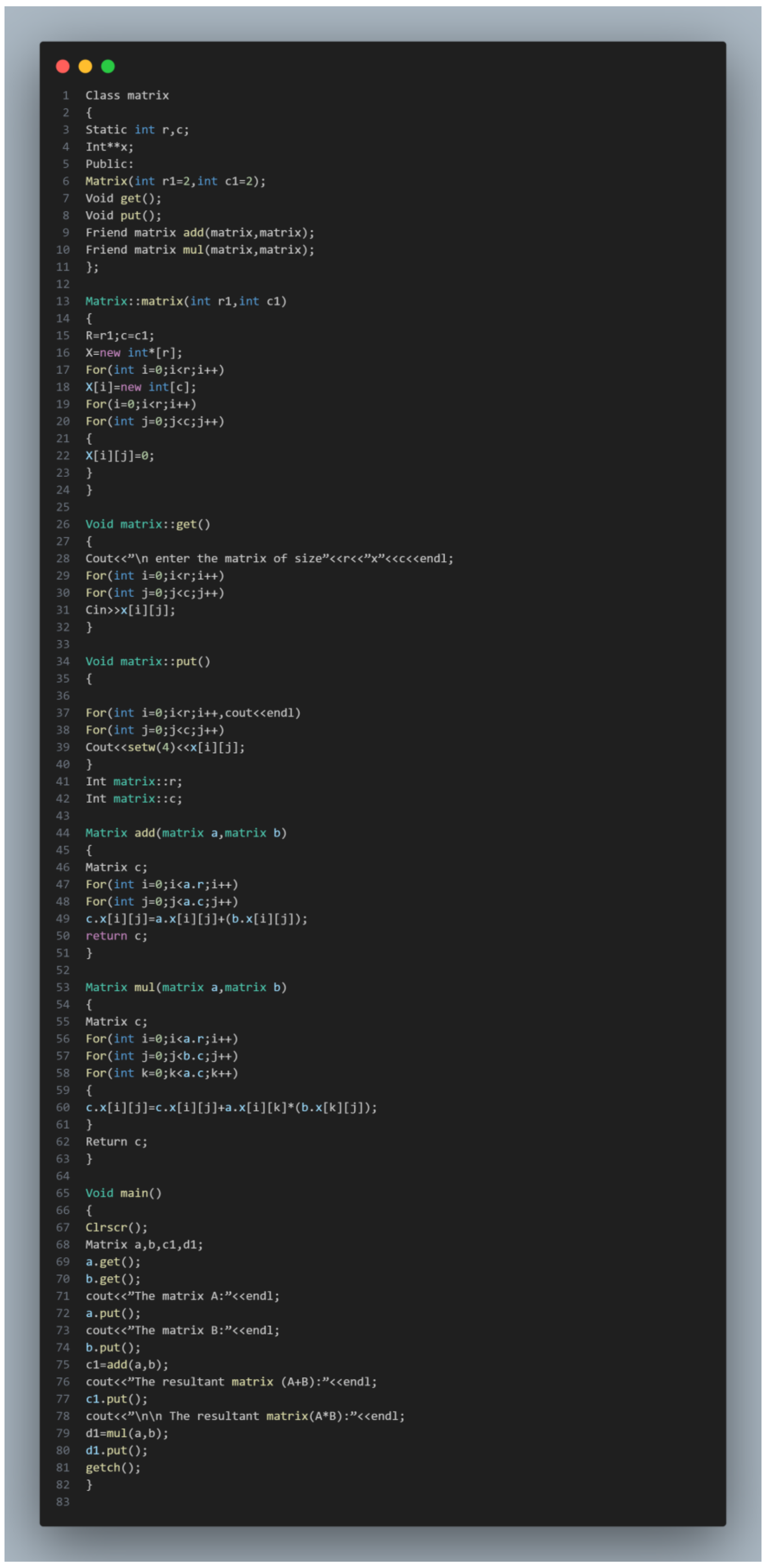
1. Write a program to create class PAPER with its properties: width & height. Find the Perimeter and Area for the objects of PAPER class using passing object as argument(s) and default arguments. (Note: Use default arguments in constructors)

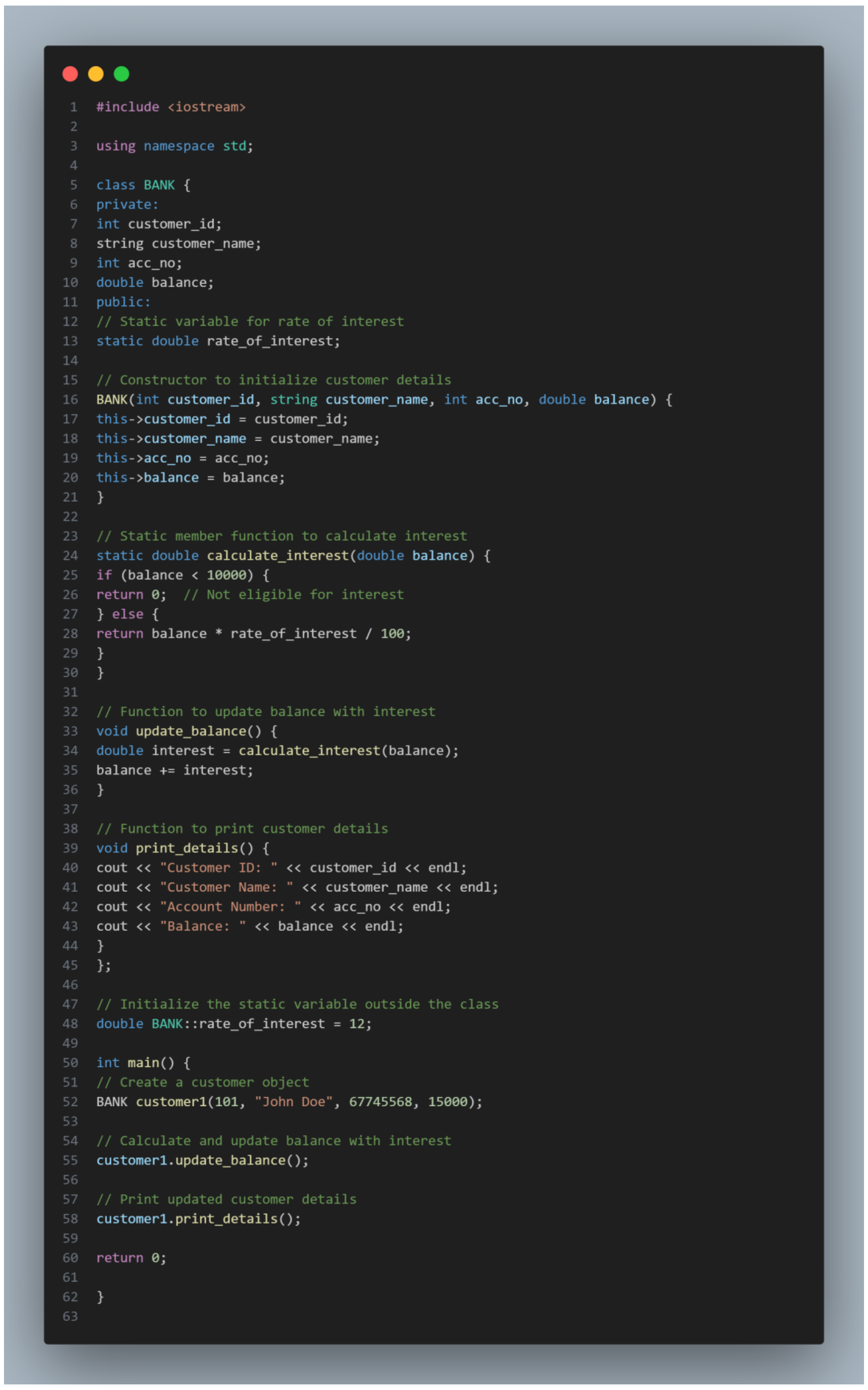
```
#include<iostream>
using namespace std;

class PAPER {
public:
    int w, h, perimeter, area;

    PAPER(int width = 10, int height = 10) {
        w = width;
        h = height;
    }

    void disp(PAPER p1) {
        perimeter = (p1.w + p1.h) * 2;
        area = p1.w * p1.h;
        cout << "\tThe perimeter of the paper is " << perimeter << " cm" << endl;
        cout << "\tThe area of the rectangle is " << area << " cm" << endl;
    }
};

int main() {
    cout << "\nDefault argument: \n";
    PAPER p1;
    p1.disp(p1);

    cout << "\n\nPassing arguments:\n";
    PAPER p2(5, 5);
    p2.disp(p2);

    return 0;
}
```

2. Write a C++ program to perform matrix manipulation using static variables, default arguments, and friend functions.



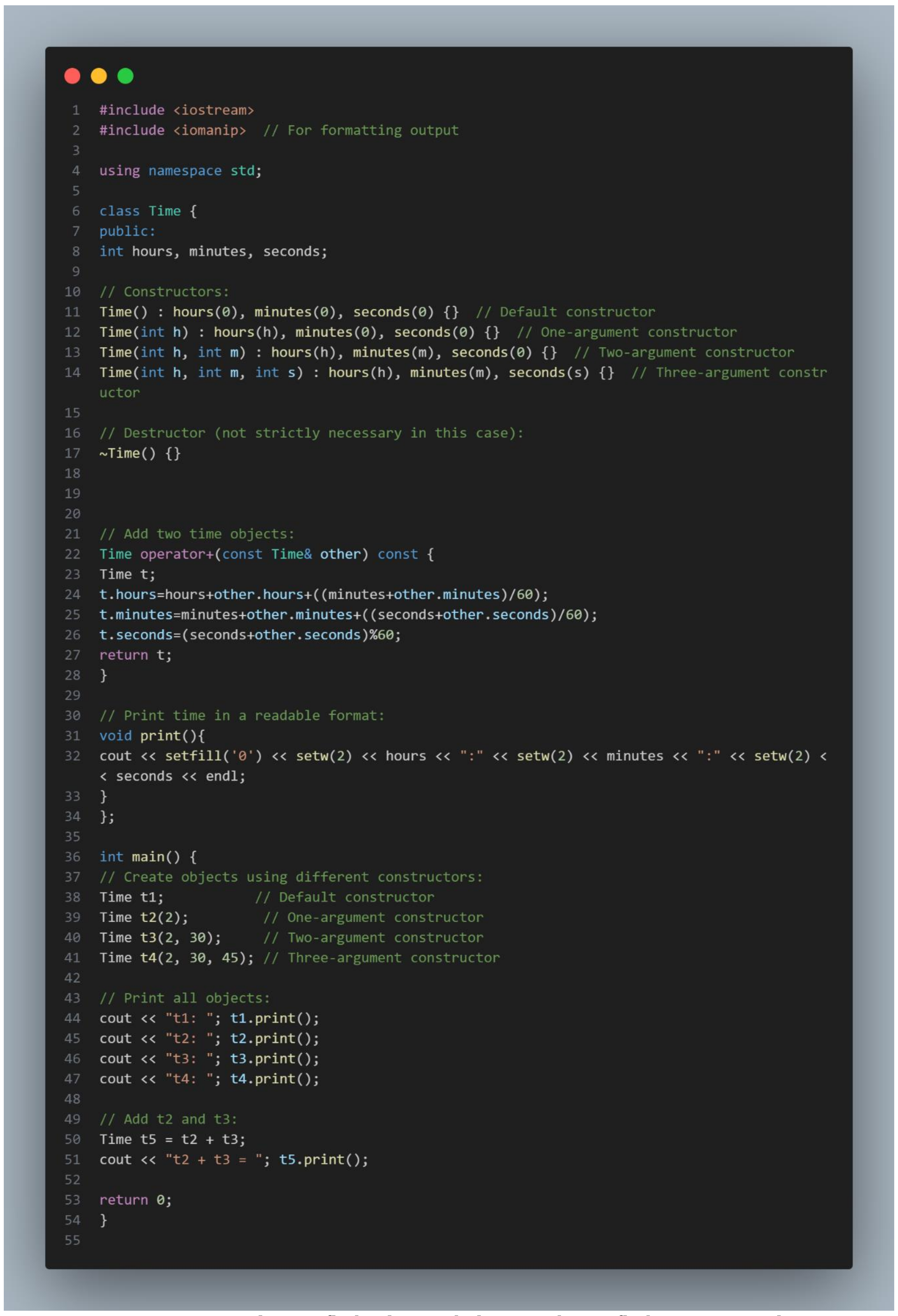
```cpp
Class matrix
{
Static int r,c;
Int**x;
Public:
Matrix(int r1=2,int c1=2);
Void get();
Void put();
Friend matrix add(matrix,matrix);
Friend matrix mul(matrix,matrix);
};

Matrix::matrix(int r1,int c1)
{
R=r1;c=c1;
X=new int*[r];
For(int i=0;i<r;i++)
X[i]=new int[c];
For(i=0;i<r;i++)
For(int j=0;j<c;j++)
{
X[i][j]=0;
}
}

Void matrix::get()
{
Cout<<"\n enter the matrix of size"<<r<<"x"<<c<<endl;
For(int i=0;i<r;i++)
For(int j=0;j<c;j++)
Cin>>x[i][j];
}

Void matrix::put()
{

For(int i=0;i<r;i++,cout<<endl)
For(int j=0;j<c;j++)
Cout<<setw(4)<<x[i][j];
}
Int matrix::r;
Int matrix::c;

Matrix add(matrix a,matrix b)
{
Matrix c;
For(int i=0;i<a.r;i++)
For(int j=0;j<a.c;j++)
c.x[i][j]=a.x[i][j]+(b.x[i][j]);
return c;
}

Matrix mul(matrix a,matrix b)
{
Matrix c;
For(int i=0;i<a.r;i++)
For(int j=0;j<b.c;j++)
For(int k=0;k<a.c;k++)
{
c.x[i][j]=c.x[i][j]+a.x[i][k]*(b.x[k][j]);
}
Return c;
}

Void main()
{
Clrscr();
Matrix a,b,c1,d1;
a.get();
b.get();
cout<<"The matrix A:"<<endl;
a.put();
cout<<"The matrix B:"<<endl;
b.put();
c1=add(a,b);
cout<<"The resultant matrix (A+B):"<<endl;
c1.put();
cout<<"\n\n The resultant matrix(A*B):"<<endl;
d1=mul(a,b);
d1.put();
getch();
}

```

3. Write a program to create a class BANK with the instance variables customer_id, customer_name, acc_no, balance and the static variable "rate_of_interest" (ROI value is 12%). Write a static member function to calculate the interest earned for the customer based on the following conditions:
   If the current balance can be in the following range:
   Less than 10000 – Not Eligible
   >=10000 – Eligible
   Add necessary constructor functions and print functions to display the updated balance of the customer.



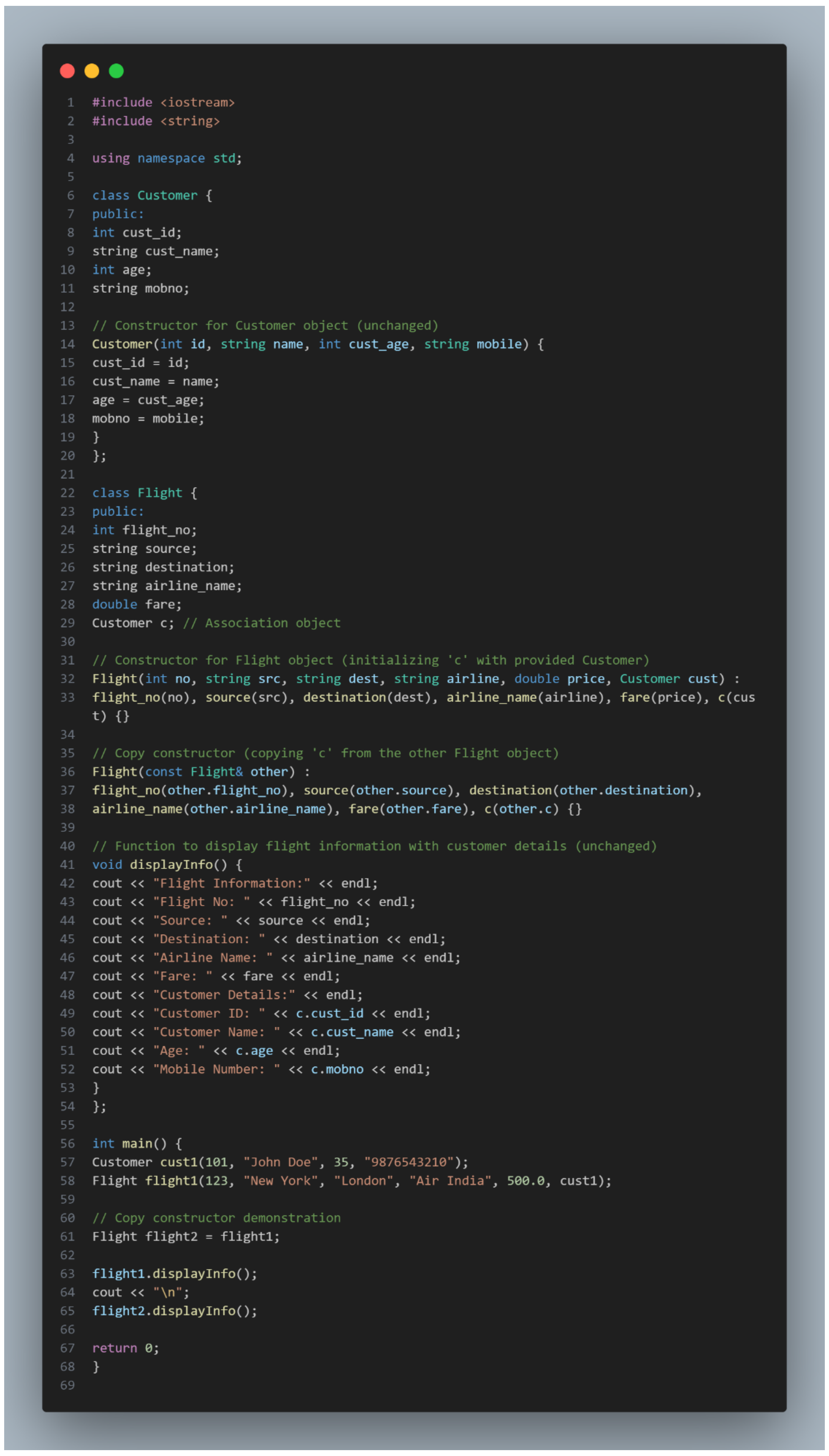
```cpp
#include <iostream>

using namespace std;

class BANK {
private:
int customer_id;
string customer_name;
int acc_no;
double balance;
public:
// Static variable for rate of interest
static double rate_of_interest;

// Constructor to initialize customer details
BANK(int customer_id, string customer_name, int acc_no, double balance) {
this->customer_id = customer_id;
this->customer_name = customer_name;
this->acc_no = acc_no;
this->balance = balance;
}

// Static member function to calculate interest
static double calculate_interest(double balance) {
if (balance < 10000) {
return 0;  // Not eligible for interest
} else {
return balance * rate_of_interest / 100;
}
}

// Function to update balance with interest
void update_balance() {
double interest = calculate_interest(balance);
balance += interest;
}

// Function to print customer details
void print_details() {
cout << "Customer ID: " << customer_id << endl;
cout << "Customer Name: " << customer_name << endl;
cout << "Account Number: " << acc_no << endl;
cout << "Balance: " << balance << endl;
}
};

// Initialize the static variable outside the class
double BANK::rate_of_interest = 12;

int main() {
// Create a customer object
BANK customer1(101, "John Doe", 67745568, 15000);

// Calculate and update balance with interest
customer1.update_balance();

// Print updated customer details
customer1.print_details();

return 0;

}
```

4. Create a “time” class which contains data members namely hours, minutes and seconds. Define default constructor, one argument, two argument, and three argument constructor and destructor methods. Also write a method to add two time values. Write a main program to invoke all the constructors and perform the addition of any two time objects and print the results.



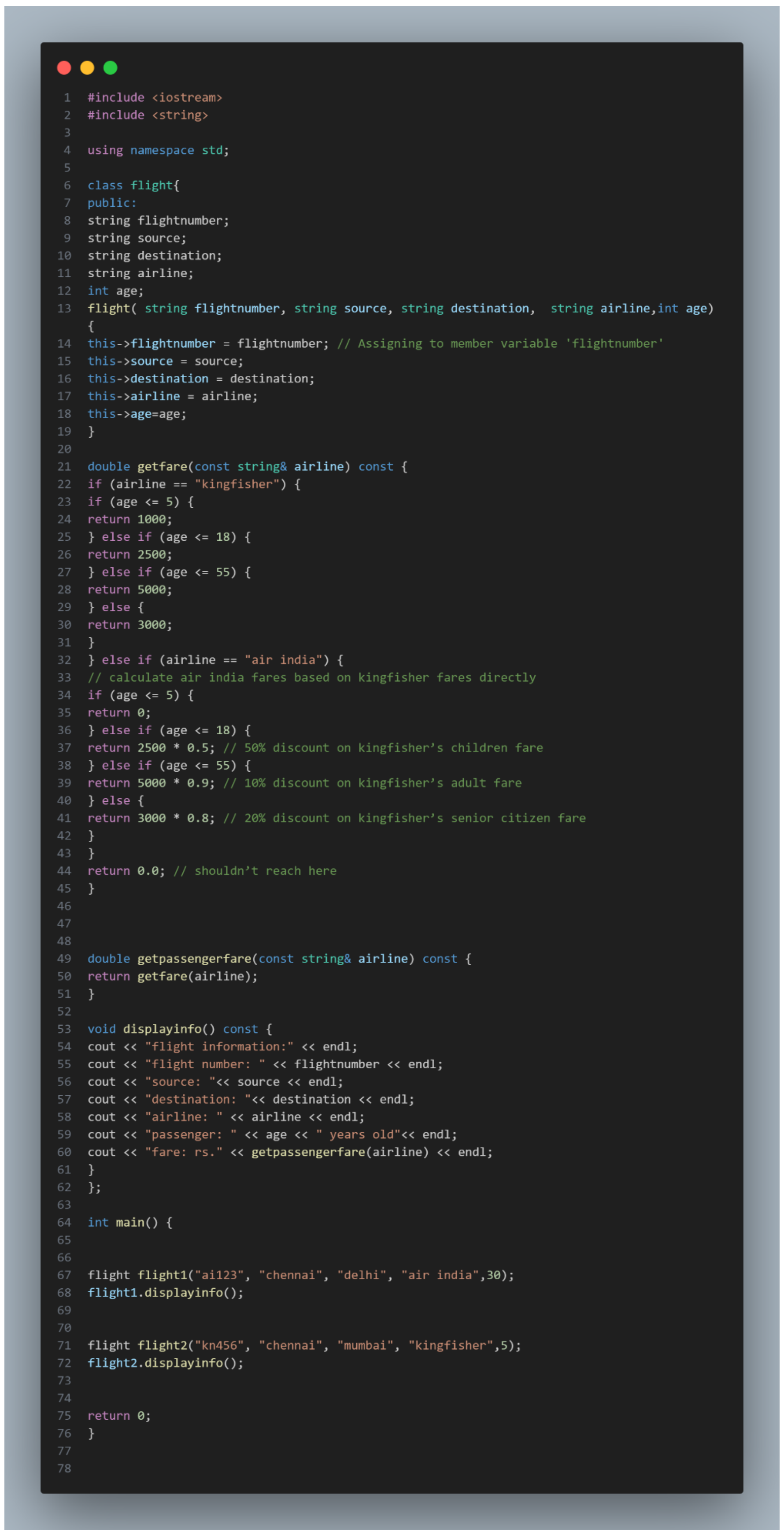
```cpp
#include <iostream>
#include <iomanip>  // For formatting output

using namespace std;

class Time {
public:
int hours, minutes, seconds;

// Constructors:
Time() : hours(0), minutes(0), seconds(0) {}  // Default constructor
Time(int h) : hours(h), minutes(0), seconds(0) {}  // One-argument constructor
Time(int h, int m) : hours(h), minutes(m), seconds(0) {}  // Two-argument constructor
Time(int h, int m, int s) : hours(h), minutes(m), seconds(s) {}  // Three-argument constructor

// Destructor (not strictly necessary in this case):
~Time() {}



// Add two time objects:
Time operator+(const Time& other) const {
Time t;
t.hours=hours+other.hours+((minutes+other.minutes)/60);
t.minutes=minutes+other.minutes+((seconds+other.seconds)/60);
t.seconds=(seconds+other.seconds)%60;
return t;
}

// Print time in a readable format:
void print(){
cout << setfill('0') << setw(2) << hours << ":" << setw(2) << minutes << ":" << setw(2) << seconds << endl;
}
};

int main() {
// Create objects using different constructors:
Time t1;           // Default constructor
Time t2(2);         // One-argument constructor
Time t3(2, 30);     // Two-argument constructor
Time t4(2, 30, 45); // Three-argument constructor

// Print all objects:
cout << "t1: "; t1.print();
cout << "t2: "; t2.print();
cout << "t3: "; t3.print();
cout << "t4: "; t4.print();

// Add t2 and t3:
Time t5 = t2 + t3;
cout << "t2 + t3 = "; t5.print();

return 0;
}

```

5. Write a C++ program to implement flight class with data member as flight no, source, destination, airline_name, fare and customer object (cust_id, cust_name, age and mobno) (use the customer object as an association object in flight class). Write a copy constructor and a member function to display the flight information with customers along with their fair details.



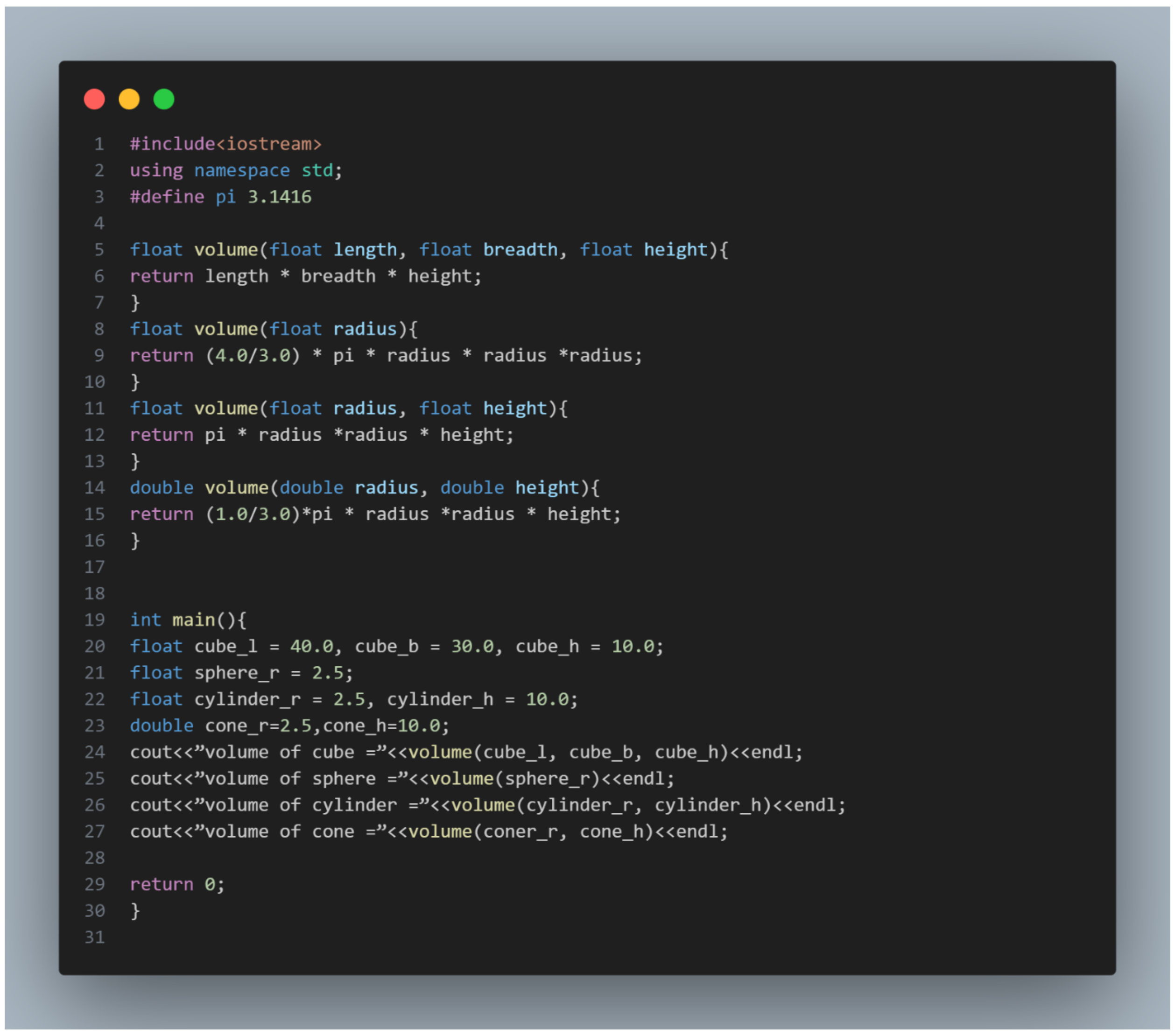
```cpp
#include <iostream>
#include <string>

using namespace std;

class Customer {
public:
int cust_id;
string cust_name;
int age;
string mobno;

// Constructor for Customer object (unchanged)
Customer(int id, string name, int cust_age, string mobile) {
cust_id = id;
cust_name = name;
age = cust_age;
mobno = mobile;
}
};

class Flight {
public:
int flight_no;
string source;
string destination;
string airline_name;
double fare;
Customer c; // Association object

// Constructor for Flight object (initializing 'c' with provided Customer)
Flight(int no, string src, string dest, string airline, double price, Customer cust) :
flight_no(no), source(src), destination(dest), airline_name(airline), fare(price), c(cus
t) {}

// Copy constructor (copying 'c' from the other Flight object)
Flight(const Flight& other) :
flight_no(other.flight_no), source(other.source), destination(other.destination),
airline_name(other.airline_name), fare(other.fare), c(other.c) {}

// Function to display flight information with customer details (unchanged)
void displayInfo() {
cout << "Flight Information:" << endl;
cout << "Flight No: " << flight_no << endl;
cout << "Source: " << source << endl;
cout << "Destination: " << destination << endl;
cout << "Airline Name: " << airline_name << endl;
cout << "Fare: " << fare << endl;
cout << "Customer Details:" << endl;
cout << "Customer ID: " << c.cust_id << endl;
cout << "Customer Name: " << c.cust_name << endl;
cout << "Age: " << c.age << endl;
cout << "Mobile Number: " << c.mobno << endl;
}
};

int main() {
Customer cust1(101, "John Doe", 35, "9876543210");
Flight flight1(123, "New York", "London", "Air India", 500.0, cust1);

// Copy constructor demonstration
Flight flight2 = flight1;

flight1.displayInfo();
cout << "\n";
flight2.displayInfo();

return 0;
}
```

6. Assume that the airlines of different company have the following plans (for seasons only):
   1.Kingfisher airlines – Source – From Chennai to any place within India.
   Fair Details: Kids (Age: 1to 5) – Rs. 1000, Children (5 to 18) – Rs.2500/-, Adults (19 to 55) – Rs. 5000/- and Senior Citizens (>55) – Rs. 3000/-. Air India - Source – From Chennai to any place within India. A common discount of 10% of the fair amount for the adult category, 20% of discount for the senior citizens and 50% of discount for the children category and free of cost for kids' category.



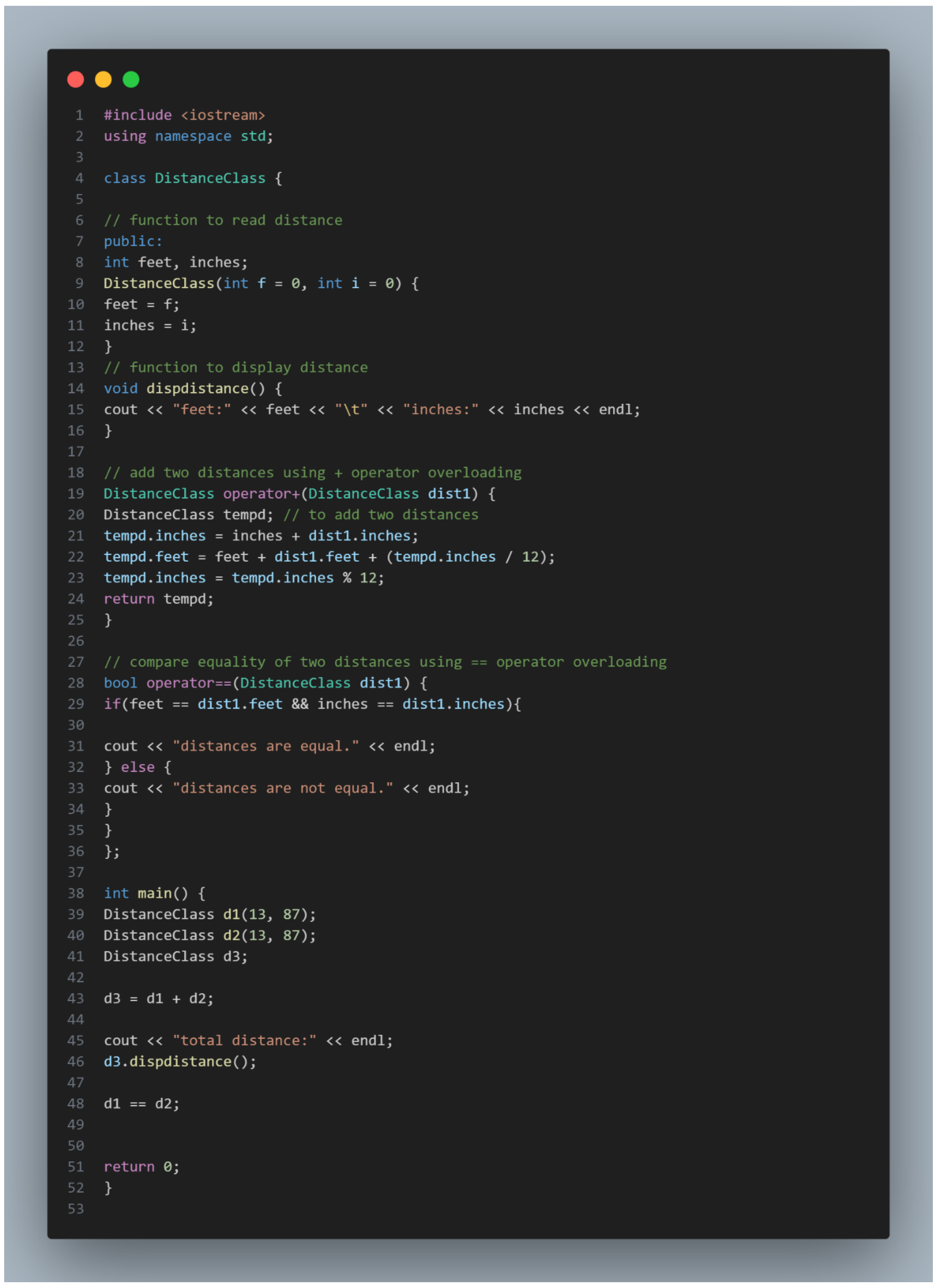
```cpp
#include <iostream>
#include <string>

using namespace std;

class flight{
public:
string flightnumber;
string source;
string destination;
string airline;
int age;
flight( string flightnumber, string source, string destination,  string airline,int age)
{
this->flightnumber = flightnumber; // Assigning to member variable 'flightnumber'
this->source = source;
this->destination = destination;
this->airline = airline;
this->age=age;
}

double getfare(const string& airline) const {
if (airline == "kingfisher") {
if (age <= 5) {
return 1000;
} else if (age <= 18) {
return 2500;
} else if (age <= 55) {
return 5000;
} else {
return 3000;
}
} else if (airline == "air india") {
// calculate air india fares based on kingfisher fares directly
if (age <= 5) {
return 0;
} else if (age <= 18) {
return 2500 * 0.5; // 50% discount on kingfisher’s children fare
} else if (age <= 55) {
return 5000 * 0.9; // 10% discount on kingfisher’s adult fare
} else {
return 3000 * 0.8; // 20% discount on kingfisher’s senior citizen fare
}
}
return 0.0; // shouldn’t reach here
}



double getpassengerfare(const string& airline) const {
return getfare(airline);
}

void displayinfo() const {
cout << "flight information:" << endl;
cout << "flight number: " << flightnumber << endl;
cout << "source: "<< source << endl;
cout << "destination: "<< destination << endl;
cout << "airline: " << airline << endl;
cout << "passenger: " << age << " years old"<< endl;
cout << "fare: rs." << getpassengerfare(airline) << endl;
}
};

int main() {


flight flight1("ai123", "chennai", "delhi", "air india",30);
flight1.displayinfo();


flight flight2("kn456", "chennai", "mumbai", "kingfisher",5);
flight2.displayinfo();


return 0;
}


```

7. Write a program to find the volume of any 4 shapes using function overloading.



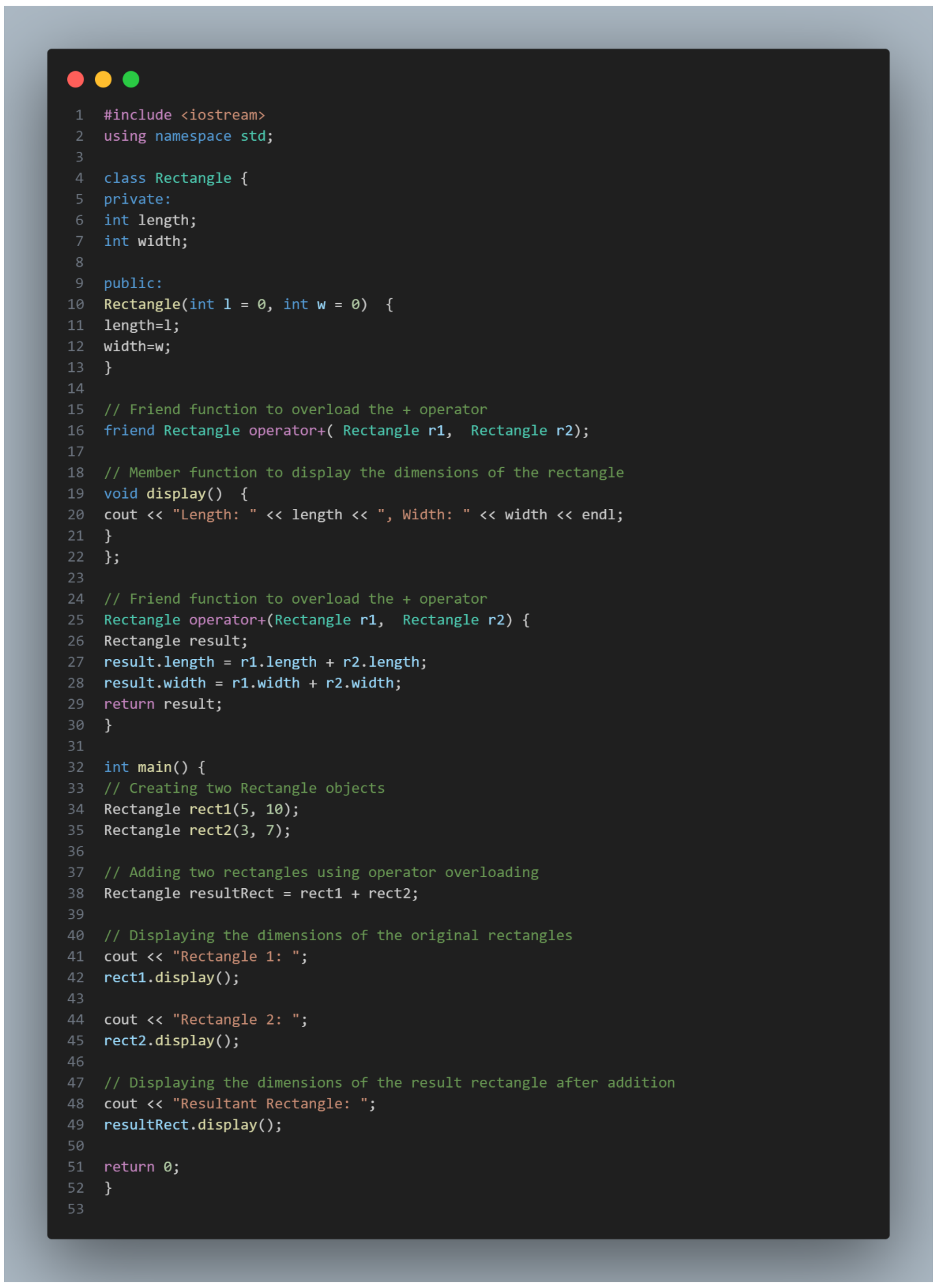
```cpp
#include<iostream>
using namespace std;
#define pi 3.1416

float volume(float length, float breadth, float height){
return length * breadth * height;
}
float volume(float radius){
return (4.0/3.0) * pi * radius * radius *radius;
}
float volume(float radius, float height){
return pi * radius *radius * height;
}
double volume(double radius, double height){
return (1.0/3.0)*pi * radius *radius * height;
}


int main(){
float cube_l = 40.0, cube_b = 30.0, cube_h = 10.0;
float sphere_r = 2.5;
float cylinder_r = 2.5, cylinder_h = 10.0;
double cone_r=2.5,cone_h=10.0;
cout<<”volume of cube =”<<volume(cube_l, cube_b, cube_h)<<endl;
cout<<”volume of sphere =”<<volume(sphere_r)<<endl;
cout<<”volume of cylinder =”<<volume(cylinder_r, cylinder_h)<<endl;
cout<<”volume of cone =”<<volume(coner_r, cone_h)<<endl;

return 0;
}
```

8. Write a program to add and compare the equality of two distance objects using operator overloading. (Note: feet and inches are the distance class attributes. Overload + and == operator)



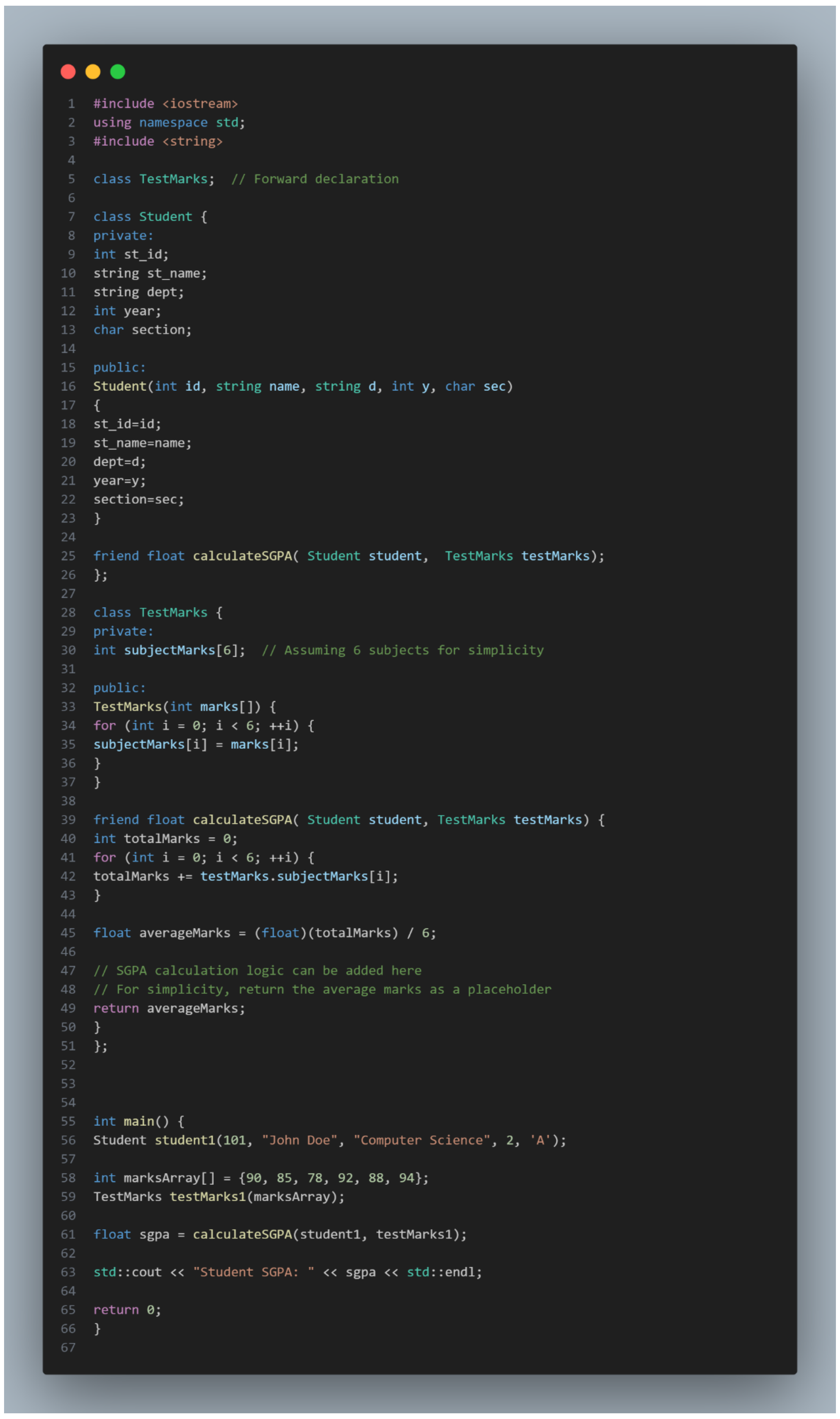
```cpp
#include <iostream>
using namespace std;

class DistanceClass {

// function to read distance
public:
int feet, inches;
DistanceClass(int f = 0, int i = 0) {
feet = f;
inches = i;
}
// function to display distance
void dispdistance() {
cout << "feet:" << feet << "\t" << "inches:" << inches << endl;
}

// add two distances using + operator overloading
DistanceClass operator+(DistanceClass dist1) {
DistanceClass tempd; // to add two distances
tempd.inches = inches + dist1.inches;
tempd.feet = feet + dist1.feet + (tempd.inches / 12);
tempd.inches = tempd.inches % 12;
return tempd;
}

// compare equality of two distances using == operator overloading
bool operator==(DistanceClass dist1) {
if(feet == dist1.feet && inches == dist1.inches){

cout << "distances are equal." << endl;
} else {
cout << "distances are not equal." << endl;
}
}
};

int main() {
DistanceClass d1(13, 87);
DistanceClass d2(13, 87);
DistanceClass d3;

d3 = d1 + d2;

cout << "total distance:" << endl;
d3.dispdistance();

d1 == d2;


return 0;
}
```

9. Write a program to add two Rectangle objects using Operator overloading with Friend Function.



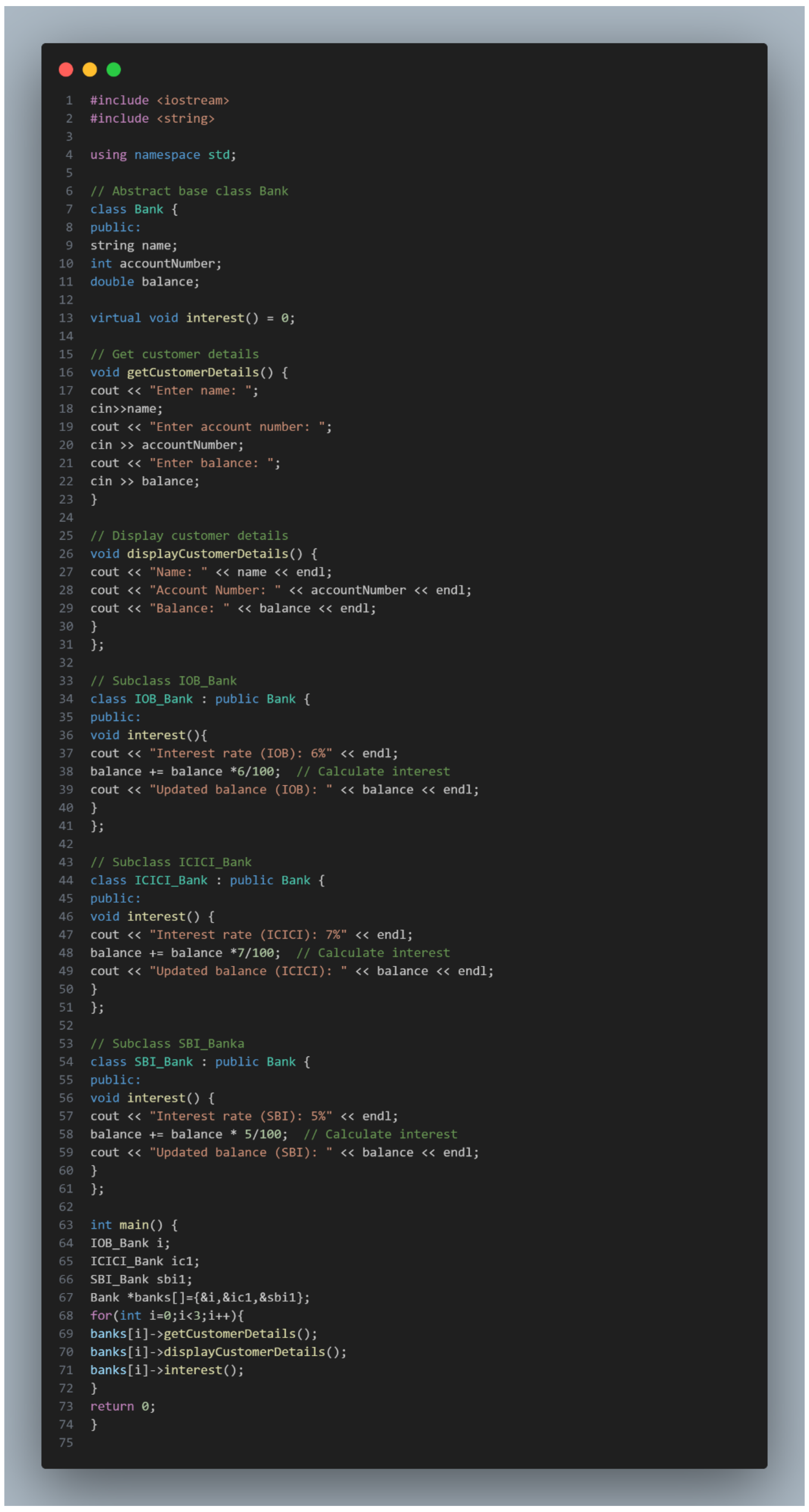
```cpp
#include <iostream>
using namespace std;

class Rectangle {
private:
int length;
int width;

public:
Rectangle(int l = 0, int w = 0)  {
length=l;
width=w;
}

// Friend function to overload the + operator
friend Rectangle operator+( Rectangle r1,  Rectangle r2);

// Member function to display the dimensions of the rectangle
void display()  {
cout << "Length: " << length << ", Width: " << width << endl;
}
};

// Friend function to overload the + operator
Rectangle operator+(Rectangle r1,  Rectangle r2) {
Rectangle result;
result.length = r1.length + r2.length;
result.width = r1.width + r2.width;
return result;
}

int main() {
// Creating two Rectangle objects
Rectangle rect1(5, 10);
Rectangle rect2(3, 7);

// Adding two rectangles using operator overloading
Rectangle resultRect = rect1 + rect2;

// Displaying the dimensions of the original rectangles
cout << "Rectangle 1: ";
rect1.display();

cout << "Rectangle 2: ";
rect2.display();

// Displaying the dimensions of the result rectangle after addition
cout << "Resultant Rectangle: ";
resultRect.display();

return 0;
}
```

10. Create a Student class with the data attributes as st_id, st_name, dept, year and section. Create another class named as TestMarks with 6 different subjects as attributes (use array to declare this subject marks). Declare both the class attributes in private access. Use Friend Classes to find the SGPA of different student objects.



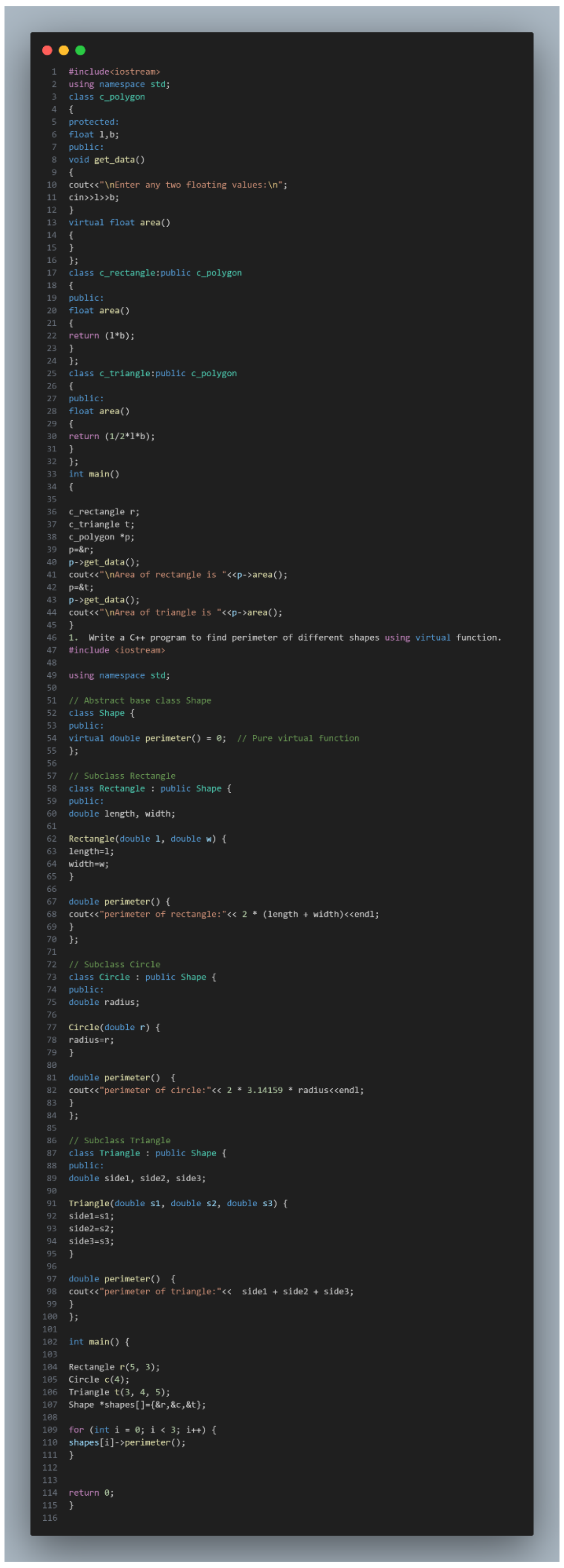
```cpp
#include <iostream>
using namespace std;
#include <string>

class TestMarks;  // Forward declaration

class Student {
private:
int st_id;
string st_name;
string dept;
int year;
char section;

public:
Student(int id, string name, string d, int y, char sec)
{
st_id=id;
st_name=name;
dept=d;
year=y;
section=sec;
}

friend float calculateSGPA( Student student,  TestMarks testMarks);
};

class TestMarks {
private:
int subjectMarks[6];  // Assuming 6 subjects for simplicity

public:
TestMarks(int marks[]) {
for (int i = 0; i < 6; ++i) {
subjectMarks[i] = marks[i];
}
}

friend float calculateSGPA( Student student, TestMarks testMarks) {
int totalMarks = 0;
for (int i = 0; i < 6; ++i) {
totalMarks += testMarks.subjectMarks[i];
}

float averageMarks = (float)(totalMarks) / 6;

// SGPA calculation logic can be added here
// For simplicity, return the average marks as a placeholder
return averageMarks;
}
};



int main() {
Student student1(101, "John Doe", "Computer Science", 2, 'A');

int marksArray[] = {90, 85, 78, 92, 88, 94};
TestMarks testMarks1(marksArray);

float sgpa = calculateSGPA(student1, testMarks1);

std::cout << "Student SGPA: " << sgpa << std::endl;

return 0;
}
```

11. Create an abstract base class “Bank” add necessary attributes and member functions in it. Define a pure virtual function named as interest () in the Bank class. Create 3 different sub classes namely IOB_Bank, ICICI_Bank and SBI_Bank. Write a C++ Program to implement the pure virtual function interest () in each of the sub class with different interest rate and display the bank customer details along with the interest rate.



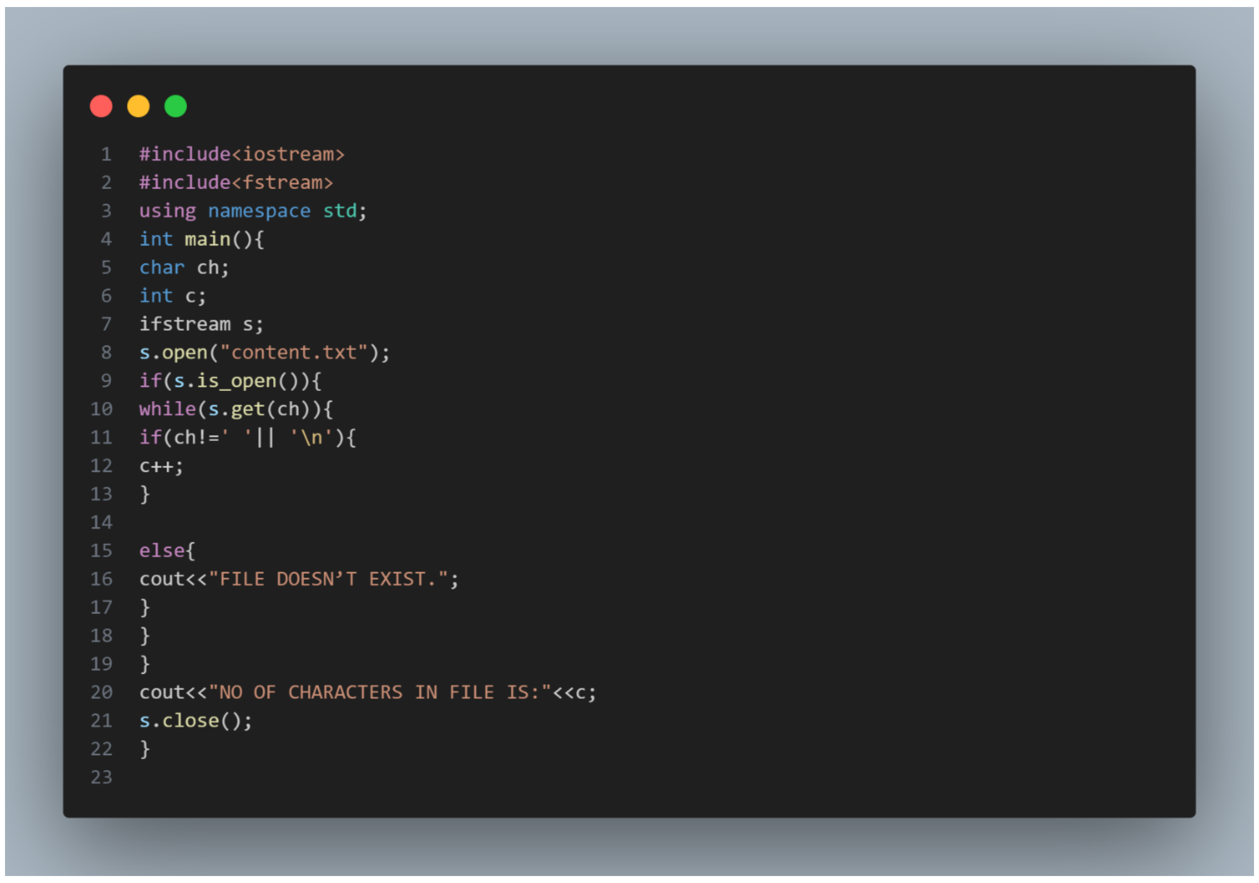
```cpp
#include <iostream>
#include <string>

using namespace std;

// Abstract base class Bank
class Bank {
public:
string name;
int accountNumber;
double balance;

virtual void interest() = 0;

// Get customer details
void getCustomerDetails() {
cout << "Enter name: ";
cin>>name;
cout << "Enter account number: ";
cin >> accountNumber;
cout << "Enter balance: ";
cin >> balance;
}

// Display customer details
void displayCustomerDetails() {
cout << "Name: " << name << endl;
cout << "Account Number: " << accountNumber << endl;
cout << "Balance: " << balance << endl;
}
};

// Subclass IOB_Bank
class IOB_Bank : public Bank {
public:
void interest(){
cout << "Interest rate (IOB): 6%" << endl;
balance += balance *6/100;  // Calculate interest
cout << "Updated balance (IOB): " << balance << endl;
}
};

// Subclass ICICI_Bank
class ICICI_Bank : public Bank {
public:
void interest() {
cout << "Interest rate (ICICI): 7%" << endl;
balance += balance *7/100;  // Calculate interest
cout << "Updated balance (ICICI): " << balance << endl;
}
};

// Subclass SBI_Banka
class SBI_Bank : public Bank {
public:
void interest() {
cout << "Interest rate (SBI): 5%" << endl;
balance += balance * 5/100;  // Calculate interest
cout << "Updated balance (SBI): " << balance << endl;
}
};

int main() {
IOB_Bank i;
ICICI_Bank ic1;
SBI_Bank sbi1;
Bank *banks[]={&i,&ic1,&sbi1};
for(int i=0;i<3;i++){
banks[i]->getCustomerDetails();
banks[i]->displayCustomerDetails();
banks[i]->interest();
}
return 0;
}
```

12. Write a C++ program to explain virtual function (polymorphism) by creating a base class c_polygon which has virtual function area (). Two classes c_rectangle and c_traingle derived from c_polygon and they have area () to calculate and return the area of rectangle and triangle respectively.



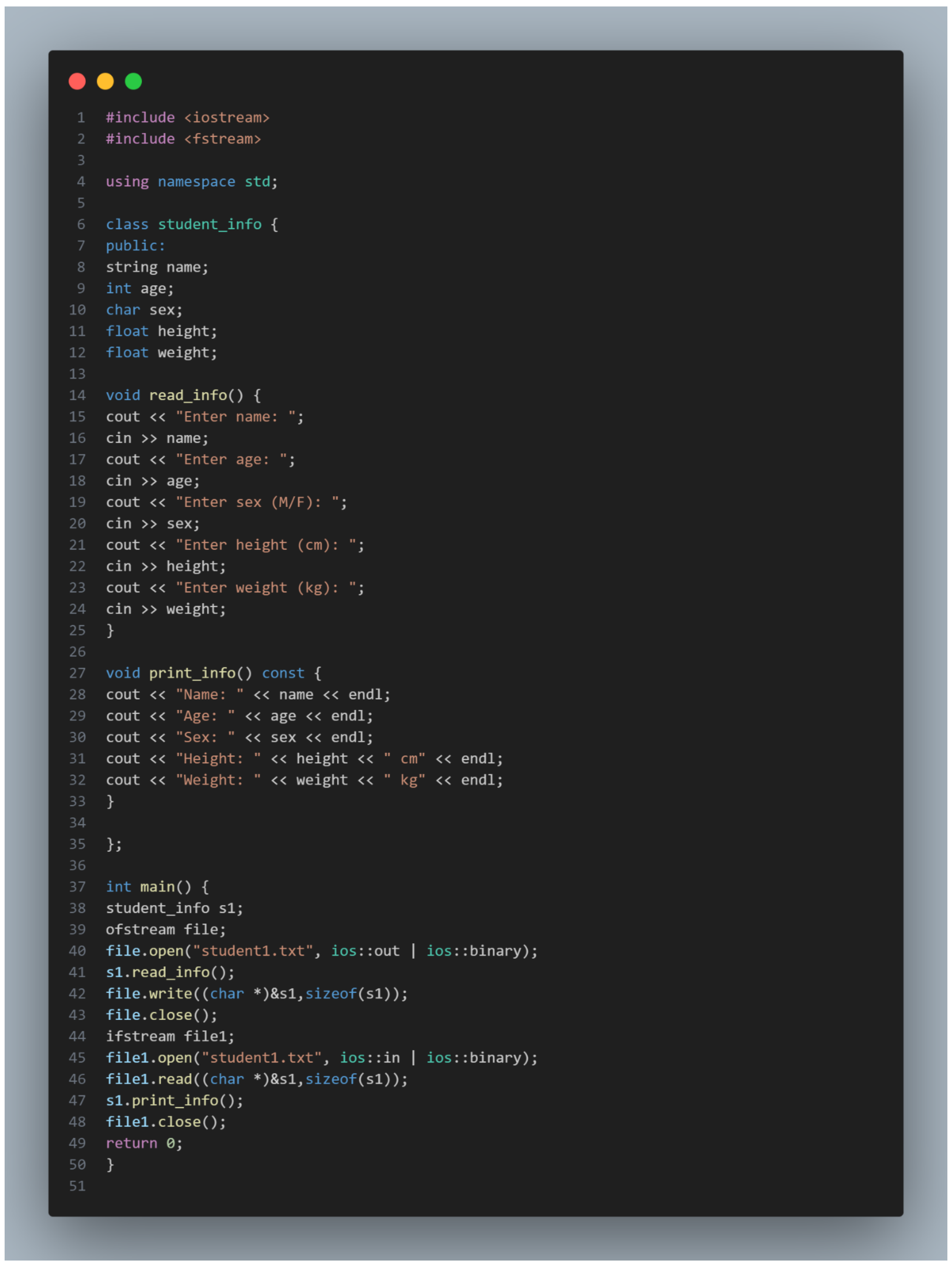
```cpp
#include<iostream>
using namespace std;
class c_polygon
{
protected:
float l,b;
public:
void get_data()
{
cout<<"\nEnter any two floating values:\n";
cin>>l>>b;
}
virtual float area()
{
}
};
class c_rectangle:public c_polygon
{
public:
float area()
{
return (l*b);
}
};
class c_triangle:public c_polygon
{
public:
float area()
{
return (1/2*l*b);
}
};
int main()
{

c_rectangle r;
c_triangle t;
c_polygon *p;
p=&r;
p->get_data();
cout<<"\nArea of rectangle is "<<p->area();
p=&t;
p->get_data();
cout<<"\nArea of triangle is "<<p->area();
}
1.  Write a C++ program to find perimeter of different shapes using virtual function.
#include <iostream>

using namespace std;

// Abstract base class Shape
class Shape {
public:
virtual double perimeter() = 0;  // Pure virtual function
};

// Subclass Rectangle
class Rectangle : public Shape {
public:
double length, width;

Rectangle(double l, double w) {
length=l;
width=w;
}

double perimeter() {
cout<<"perimeter of rectangle:"<< 2 * (length + width)<<endl;
}
};

// Subclass Circle
class Circle : public Shape {
public:
double radius;

Circle(double r) {
radius=r;
}

double perimeter()  {
cout<<"perimeter of circle:"<< 2 * 3.14159 * radius<<endl;
}
};

// Subclass Triangle
class Triangle : public Shape {
public:
double side1, side2, side3;

Triangle(double s1, double s2, double s3) {
side1=s1;
side2=s2;
side3=s3;
}

double perimeter()  {
cout<<"perimeter of triangle:"<<  side1 + side2 + side3;
}
};

int main() {

Rectangle r(5, 3);
Circle c(4);
Triangle t(3, 4, 5);
Shape *shapes[]={&r,&c,&t};

for (int i = 0; i < 3; i++) {
shapes[i]->perimeter();
}


return 0;
}

```

13. Write a program to count the number of characters in a file and display the total number of characters in console.



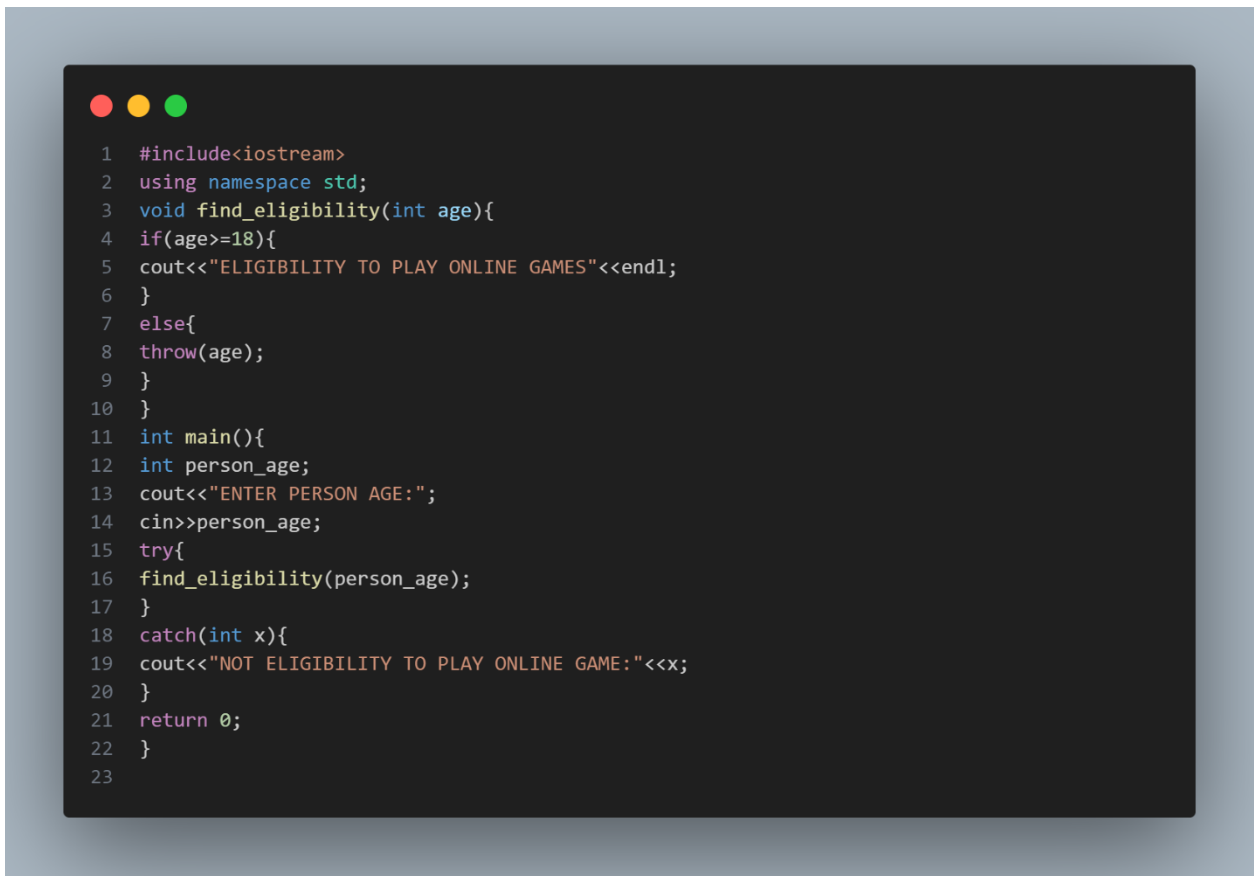
```cpp
#include<iostream>
#include<fstream>
using namespace std;
int main(){
char ch;
int c;
ifstream s;
s.open("content.txt");
if(s.is_open()){
while(s.get(ch)){
if(ch!=' '|| '\n'){
c++;
}

else{
cout<<"FILE DOESN'T EXIST.";
}
}
}
cout<<"NO OF CHARACTERS IN FILE IS:"<<c;
s.close();
}

```

14. Write a program to read the class object of student_info such as name, age, sex, height, and weight from the keyboard and to store them on a specified file using read () and write () functions. Again, the same file is opened for reading and displaying the contents of the file on the screen.



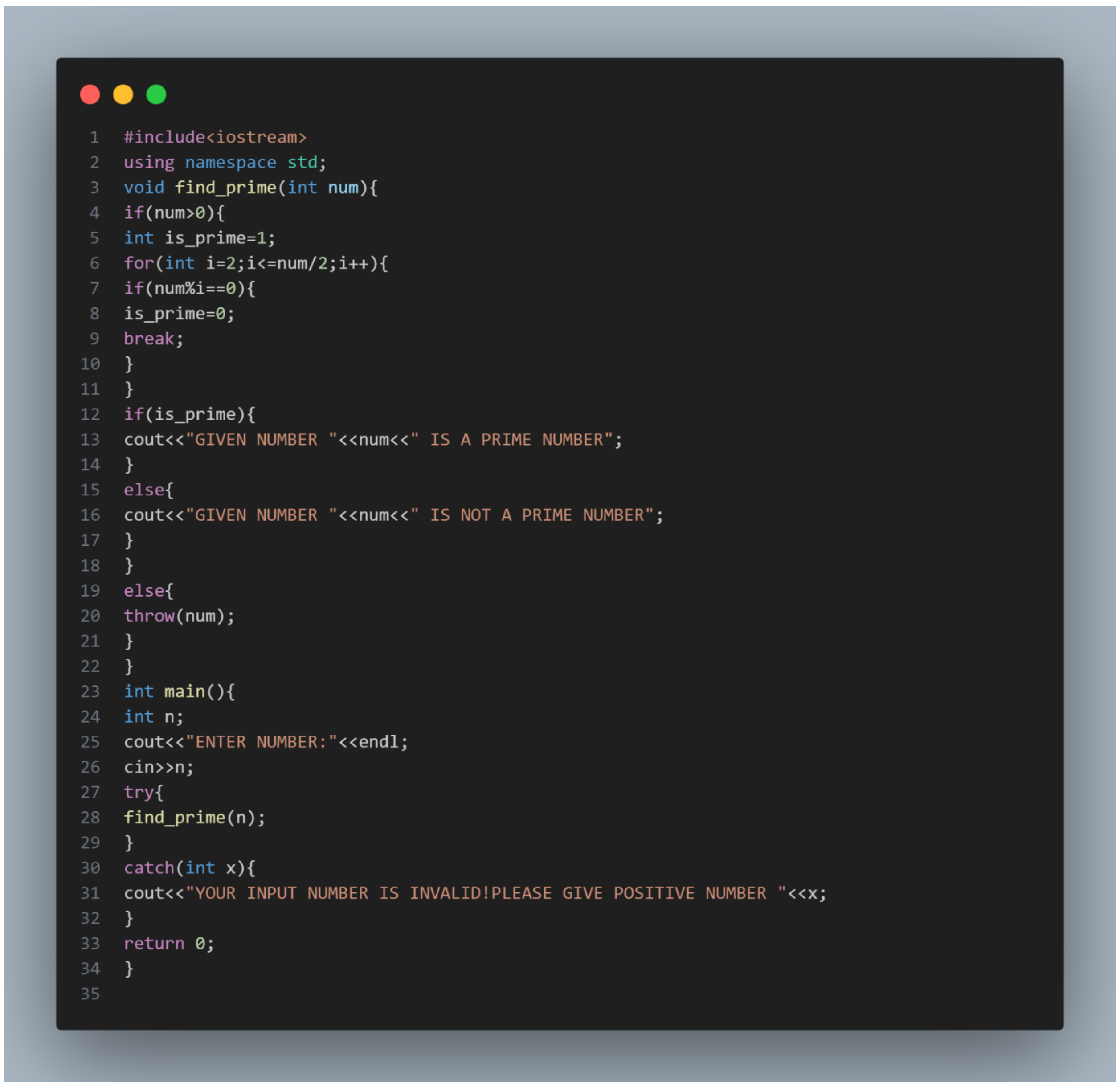
```cpp
#include <iostream>
#include <fstream>

using namespace std;

class student_info {
public:
string name;
int age;
char sex;
float height;
float weight;

void read_info() {
cout << "Enter name: ";
cin >> name;
cout << "Enter age: ";
cin >> age;
cout << "Enter sex (M/F): ";
cin >> sex;
cout << "Enter height (cm): ";
cin >> height;
cout << "Enter weight (kg): ";
cin >> weight;
}

void print_info() const {
cout << "Name: " << name << endl;
cout << "Age: " << age << endl;
cout << "Sex: " << sex << endl;
cout << "Height: " << height << " cm" << endl;
cout << "Weight: " << weight << " kg" << endl;
}

};

int main() {
student_info s1;
ofstream file;
file.open("student1.txt", ios::out | ios::binary);
s1.read_info();
file.write((char *)&s1,sizeof(s1));
file.close();
ifstream file1;
file1.open("student1.txt", ios::in | ios::binary);
file1.read((char *)&s1,sizeof(s1));
s1.print_info();
file1.close();
return 0;
}
```

15. Write a program to check the eligibility criteria of a person to play the online game based on age criteria. Raise an exception with an appropriate message if the criterion is not satisfied.



```
#include<iostream>
using namespace std;
void find_eligibility(int age){
if(age>=18){
cout<<"ELIGIBILITY TO PLAY ONLINE GAMES"<<endl;
}
else{
throw(age);
}
}
int main(){
int person_age;
cout<<"ENTER PERSON AGE:";
cin>>person_age;
try{
find_eligibility(person_age);
}
catch(int x){
cout<<"NOT ELIGIBILITY TO PLAY ONLINE GAME:"<<x;
}
return 0;
}
```

17.Write a program to find whether a given input number is a prime number or not. Throw exceptions if the user enters zero or negative value as input.



```cpp
#include<iostream>
using namespace std;
void find_prime(int num){
if(num>0){
int is_prime=1;
for(int i=2;i<=num/2;i++){
if(num%i==0){
is_prime=0;
break;
}
}
if(is_prime){
cout<<"GIVEN NUMBER "<<num<<" IS A PRIME NUMBER";
}
else{
cout<<"GIVEN NUMBER "<<num<<" IS NOT A PRIME NUMBER";
}
}
else{
throw(num);
}
}
int main(){
int n;
cout<<"ENTER NUMBER:"<<endl;
cin>>n;
try{
find_prime(n);
}
catch(int x){
cout<<"YOUR INPUT NUMBER IS INVALID!PLEASE GIVE POSITIVE NUMBER "<<x;
}
return 0;
}

```