PRACTICAL

LIBRARY MANAGEMENT SCENARIO.

EMPLOYEE PAYROLL SYSTEM SCENARIO.

LIBRARY MANAGEMENT SYSTEM:

- *Efficient*: The system should be able to quickly process book additions, issuances, and returns.

- *Organized*: The system should store book data in a structured and easily accessible manner.

- *Persistent*: The system should store book data in a file for long-term storage.

- *User-friendly*: The system should provide an intuitive interface for users to interact with.

- *Informative*: The system should display relevant statistics and information about the book collection.

PROBLEM:

- 1. Library Management System
- Scenario:
- Design a system to manage a library’s book collection. The program should allow users to add new books, issue books to students, and track the return of borrowed books.
- Requirements:
- Create a Book class with attributes like book ID, title, author, and availability status.
- Implement methods to issue and return books.
- Design a Library class to store a collection of books and provide a method to search for a book by title or ID.
- Display statistics such as the total number of books, issued books, and available books.
- Store the book data in a file for persistent storage.

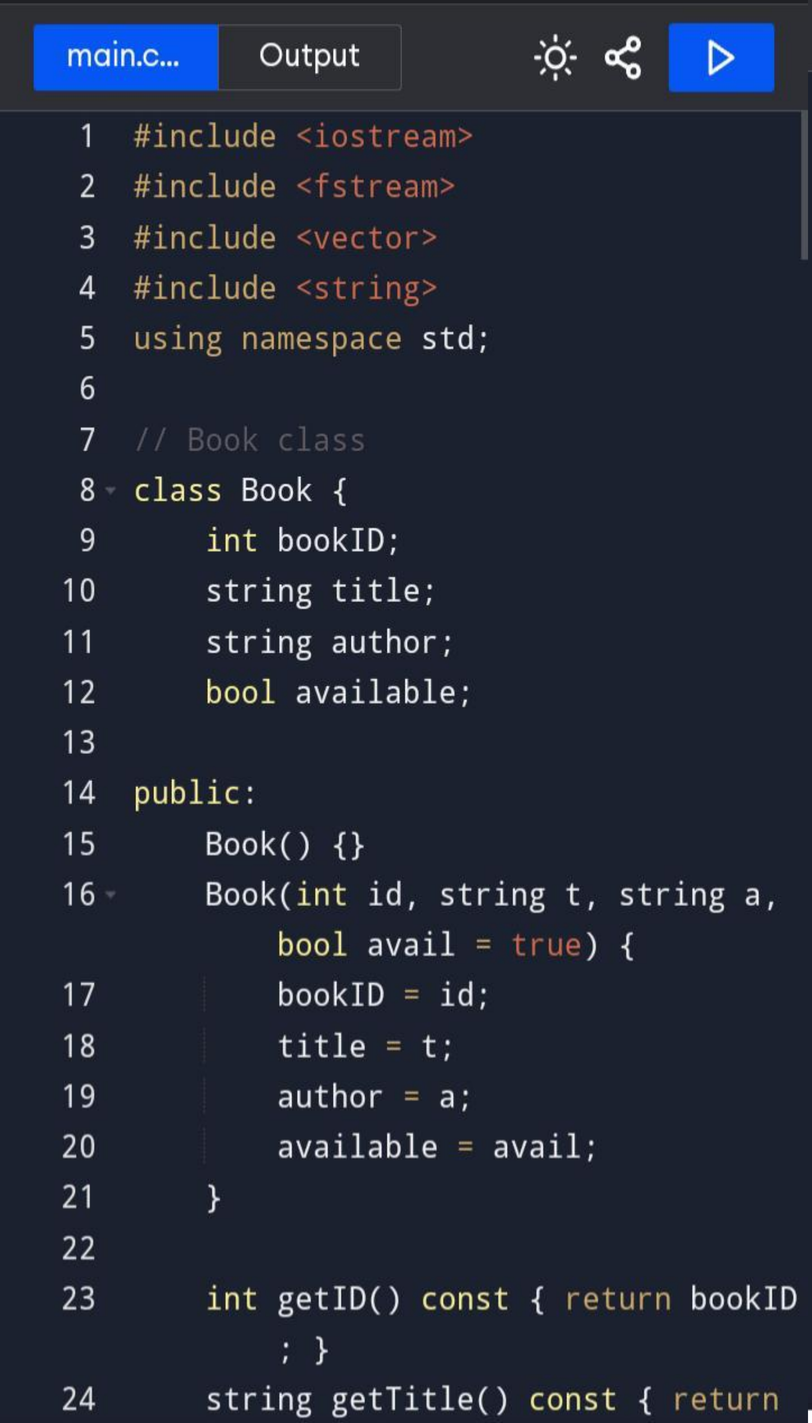
1. The system can be extended to manage student records, linking each issued book with the student ID.

2. A due date and fine calculation feature could be added for overdue books.

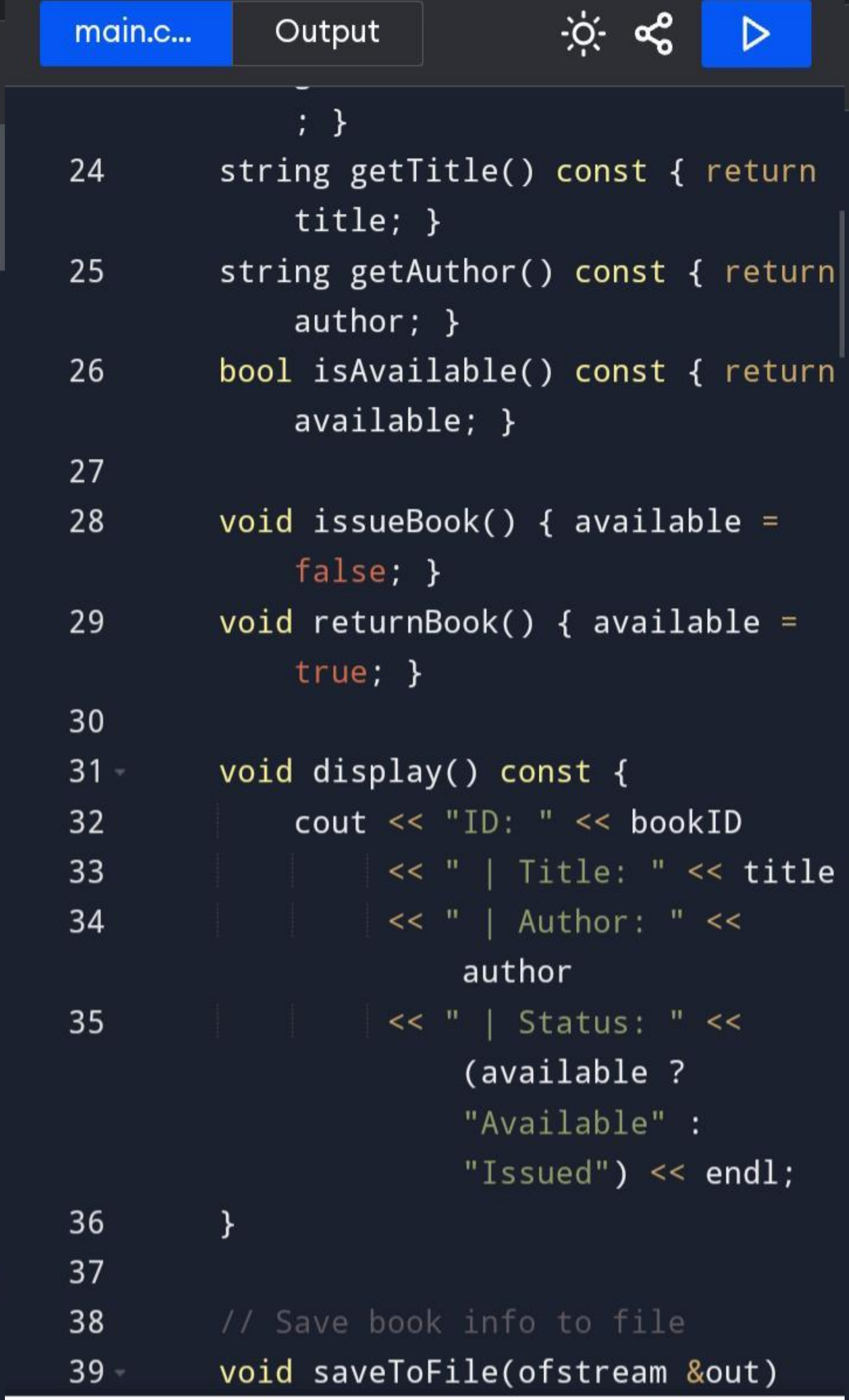
3. The file storage can be upgraded to a database for faster searching and better scalability.

4. The system may include a feature to categorize books by subject or genre.

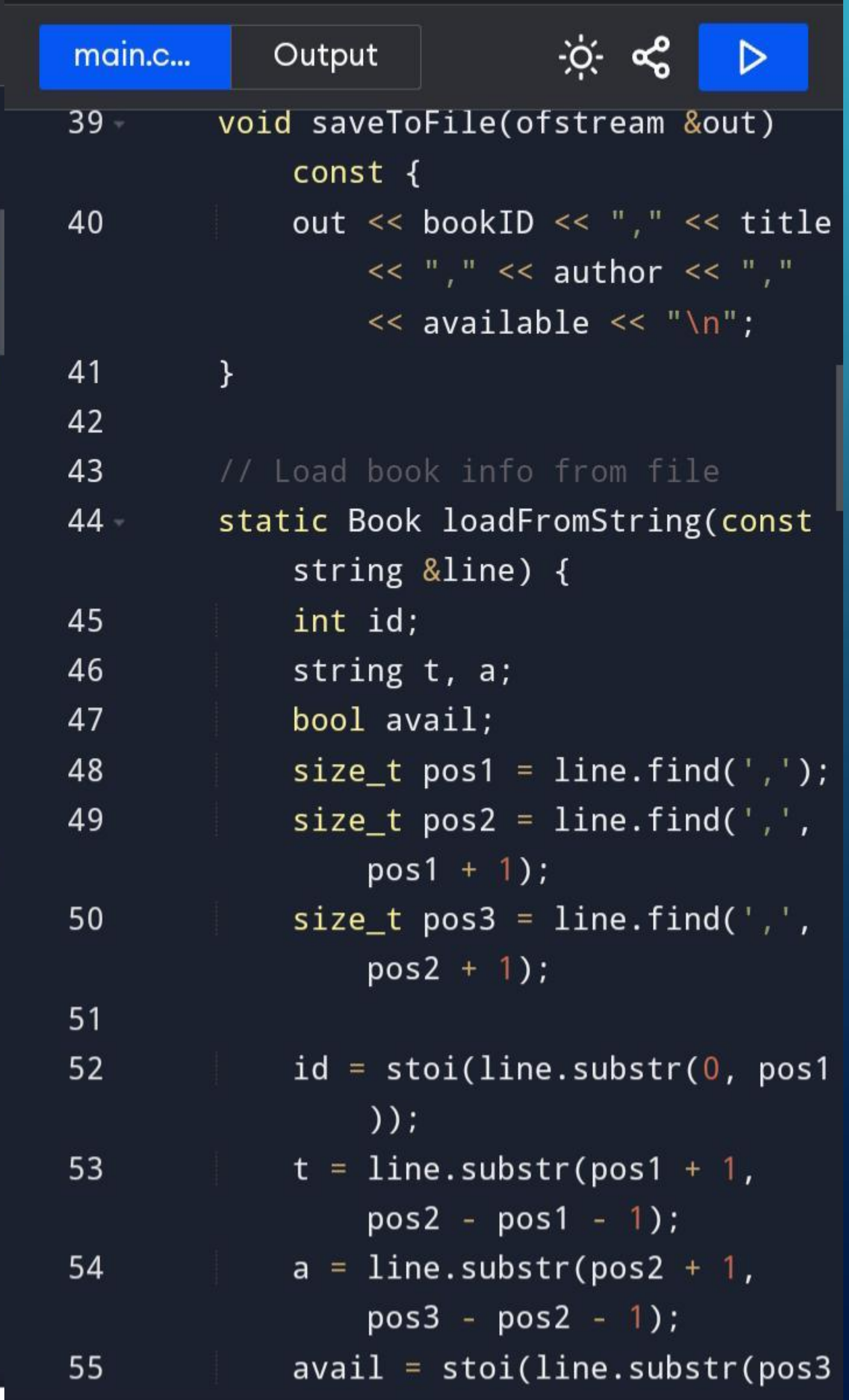
5. A graphical user interface (GUI)



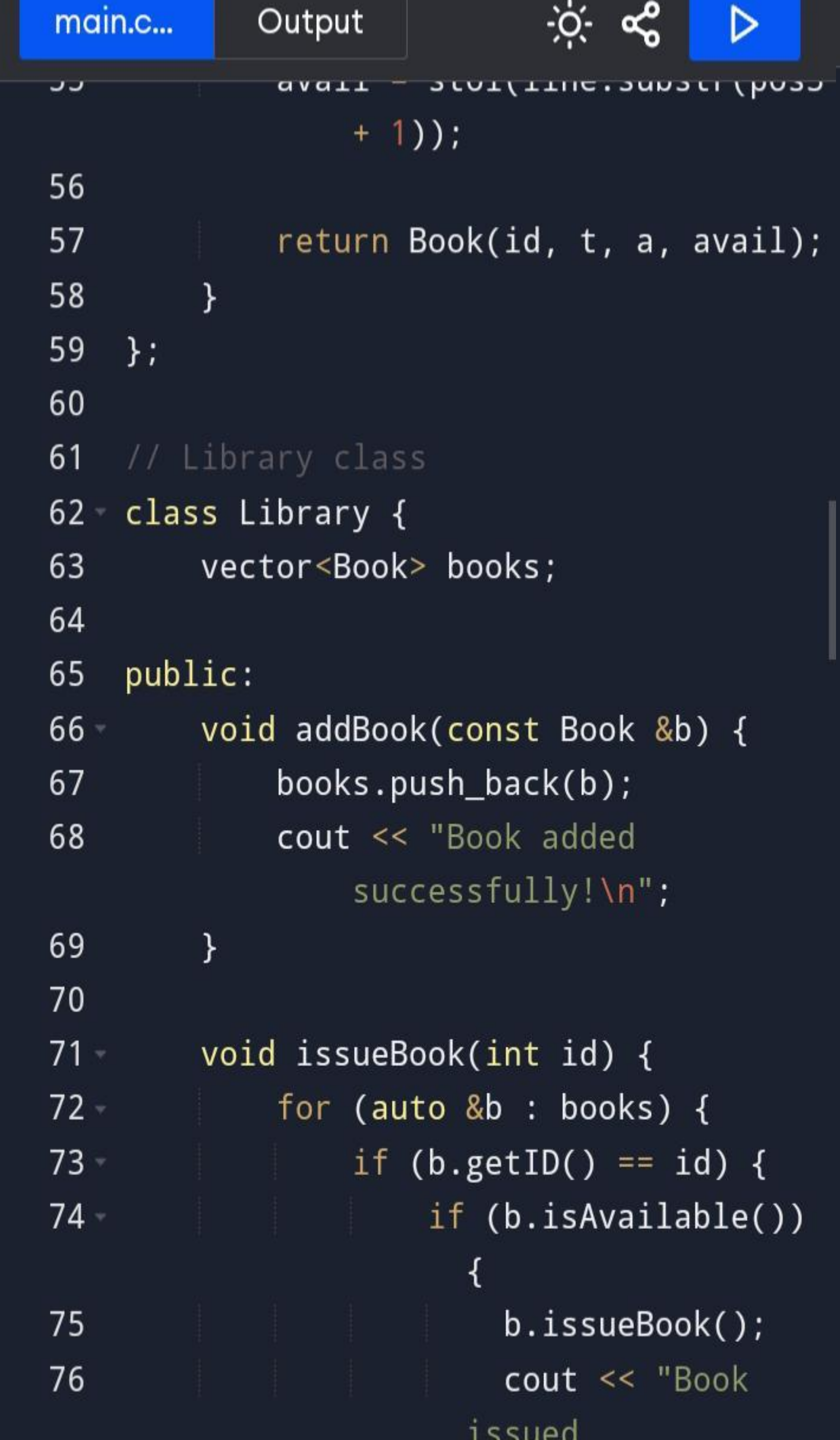
```cpp
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
using namespace std;

// Book class
class Book {
    int bookID;
    string title;
    string author;
    bool available;

public:
    Book() {}
    Book(int id, string t, string a, bool avail = true) {
        bookID = id;
        title = t;
        author = a;
        available = avail;
    }

    int getID() const { return bookID; }
    string getTitle() const { return
```



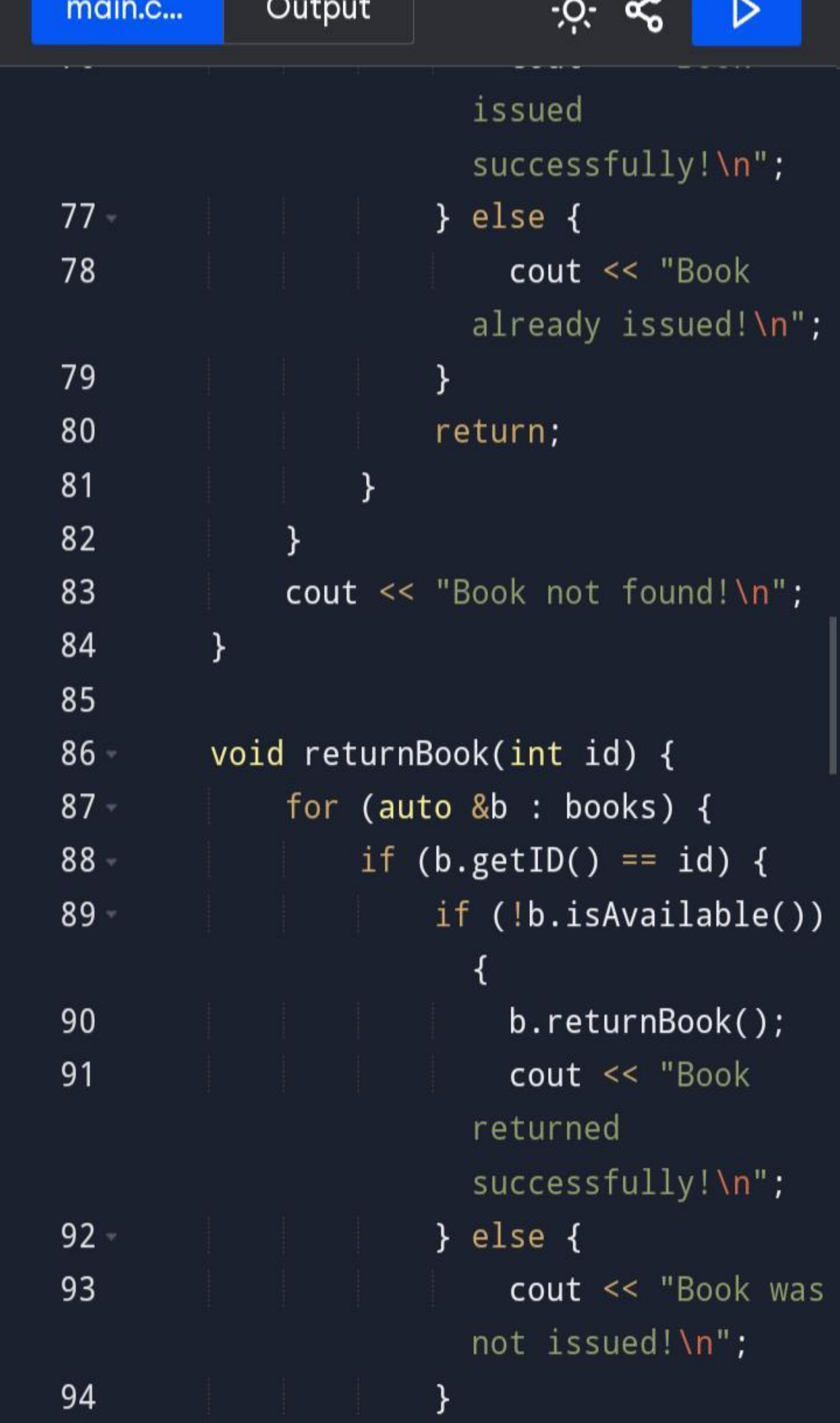
```cpp
        ; }
    string getTitle() const { return title; }
    string getAuthor() const { return author; }
    bool isAvailable() const { return available; }

    void issueBook() { available = false; }
    void returnBook() { available = true; }

    void display() const {
        cout << "ID: " << bookID
             << " | Title: " << title
             << " | Author: " << author
             << " | Status: " << (available ? "Available" : "Issued") << endl;
    }

    // Save book info to file
    void saveToFile(ofstream &out)
```



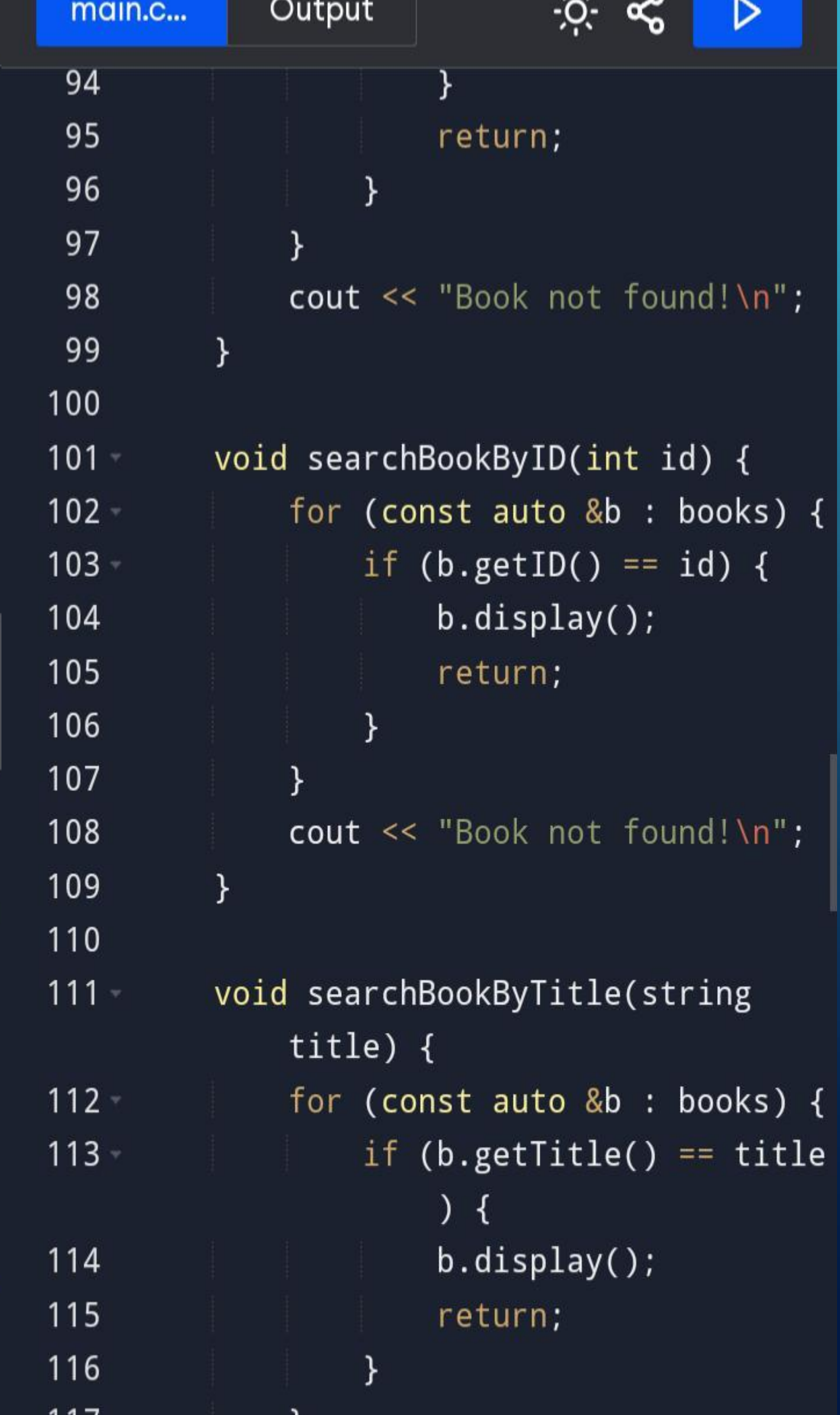
```cpp
    void saveToFile(ofstream &out) const {
        out << bookID << "," << title << "," << author << "," << available << "\n";
    }

    // Load book info from file
    static Book loadFromString(const string &line) {
        int id;
        string t, a;
        bool avail;
        size_t pos1 = line.find(',');
        size_t pos2 = line.find(',', pos1 + 1);
        size_t pos3 = line.find(',', pos2 + 1);

        id = stoi(line.substr(0, pos1));
        t = line.substr(pos1 + 1, pos2 - pos1 - 1);
        a = line.substr(pos2 + 1, pos3 - pos2 - 1);
        avail = stoi(line.substr(pos3
```



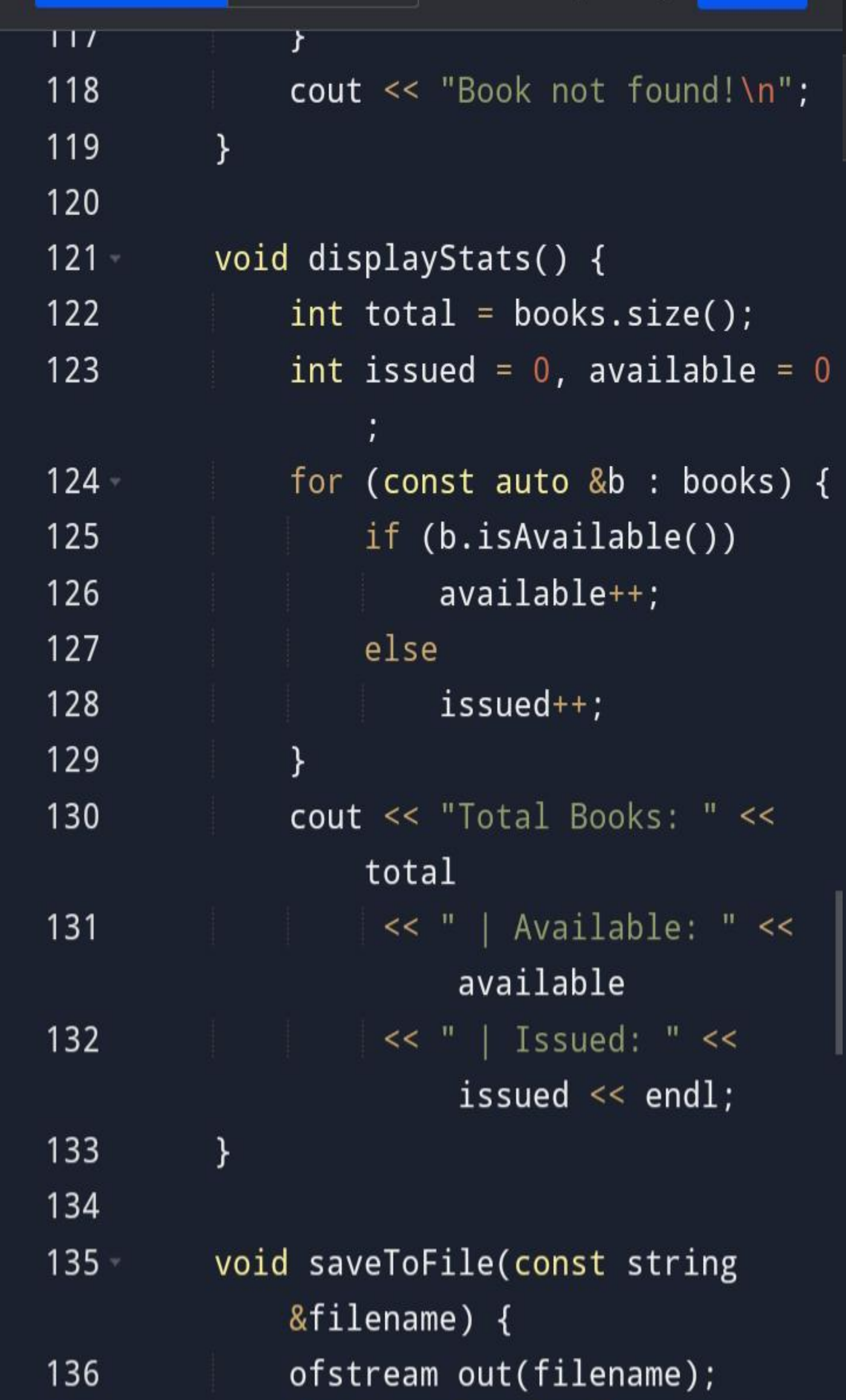
```cpp
        avail = stoi(line.substr(pos3 + 1));

        return Book(id, t, a, avail);
    }
};

// Library class
class Library {
    vector<Book> books;

public:
    void addBook(const Book &b) {
        books.push_back(b);
        cout << "Book added successfully!\n";
    }

    void issueBook(int id) {
        for (auto &b : books) {
            if (b.getID() == id) {
                if (b.isAvailable()) {
                    b.issueBook();
                    cout << "Book issued
```



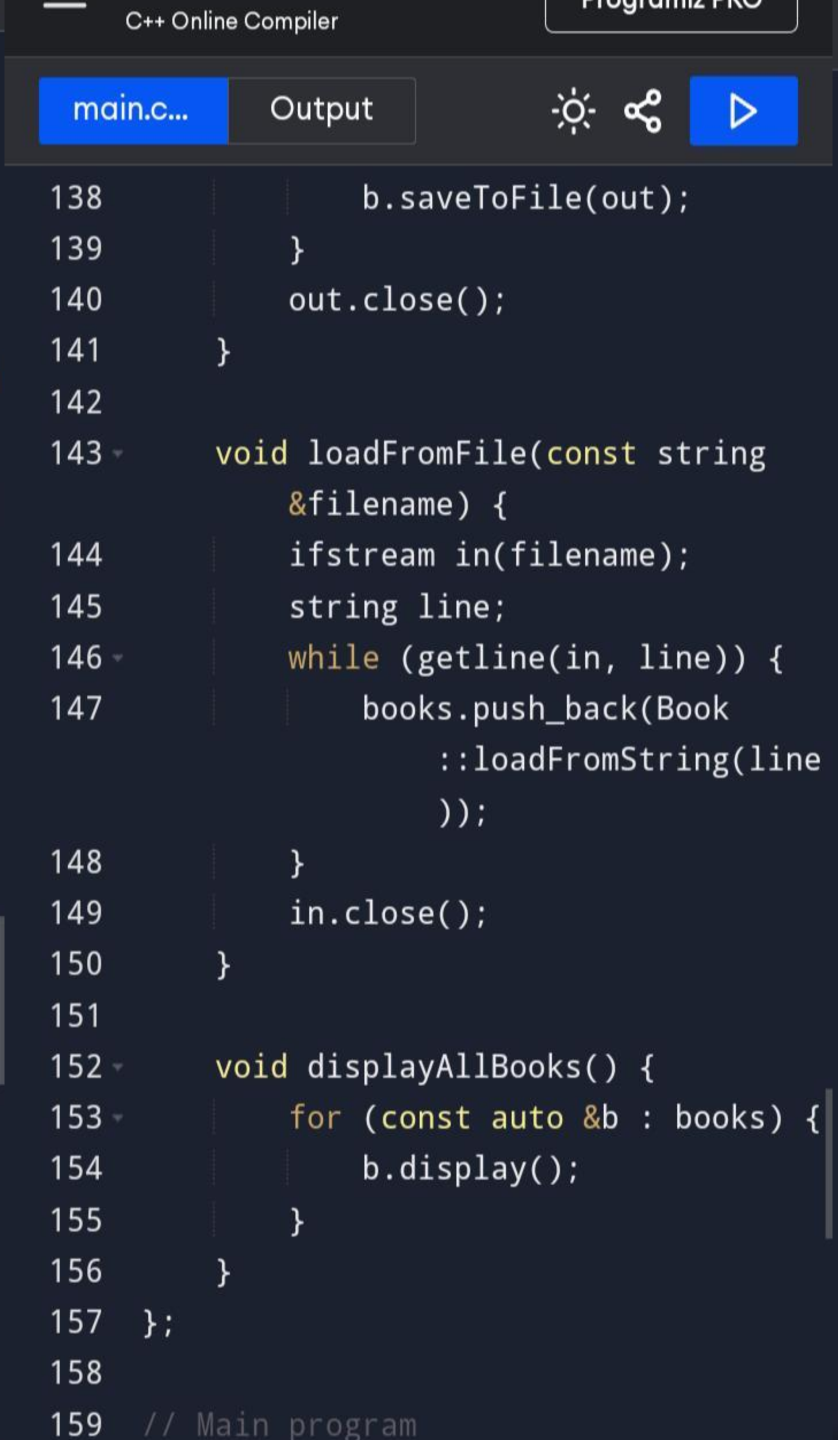
```cpp
                    cout << "Book issued successfully!\n";
                } else {
                    cout << "Book already issued!\n";
                }
                return;
            }
        }
        cout << "Book not found!\n";
    }

    void returnBook(int id) {
        for (auto &b : books) {
            if (b.getID() == id) {
                if (!b.isAvailable()) {
                    b.returnBook();
                    cout << "Book returned successfully!\n";
                } else {
                    cout << "Book was not issued!\n";
                }
```



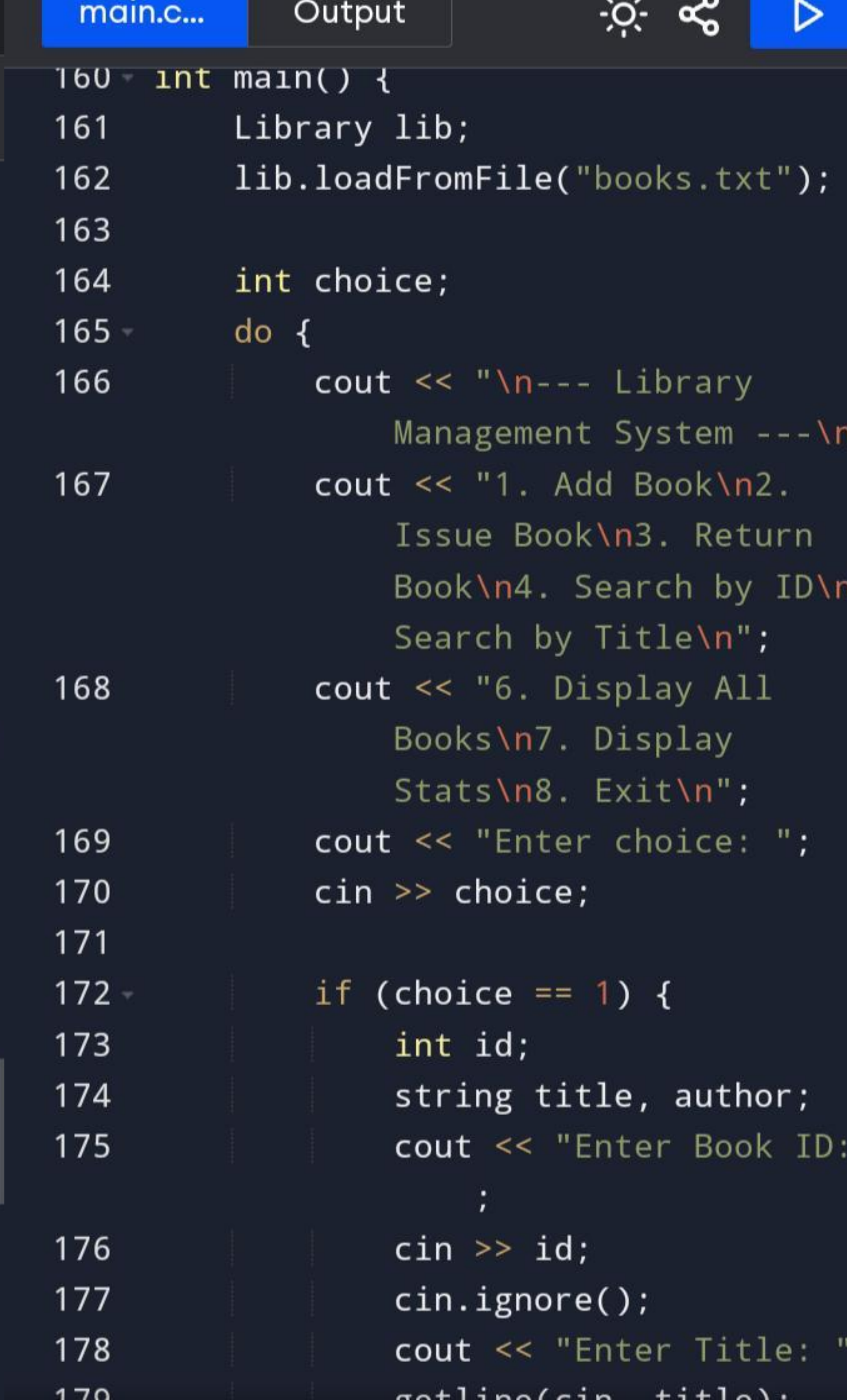
```cpp
                }
                return;
            }
        }
        cout << "Book not found!\n";
    }

    void searchBookByID(int id) {
        for (const auto &b : books) {
            if (b.getID() == id) {
                b.display();
                return;
            }
        }
        cout << "Book not found!\n";
    }

    void searchBookByTitle(string title) {
        for (const auto &b : books) {
            if (b.getTitle() == title) {
                b.display();
                return;
            }
```



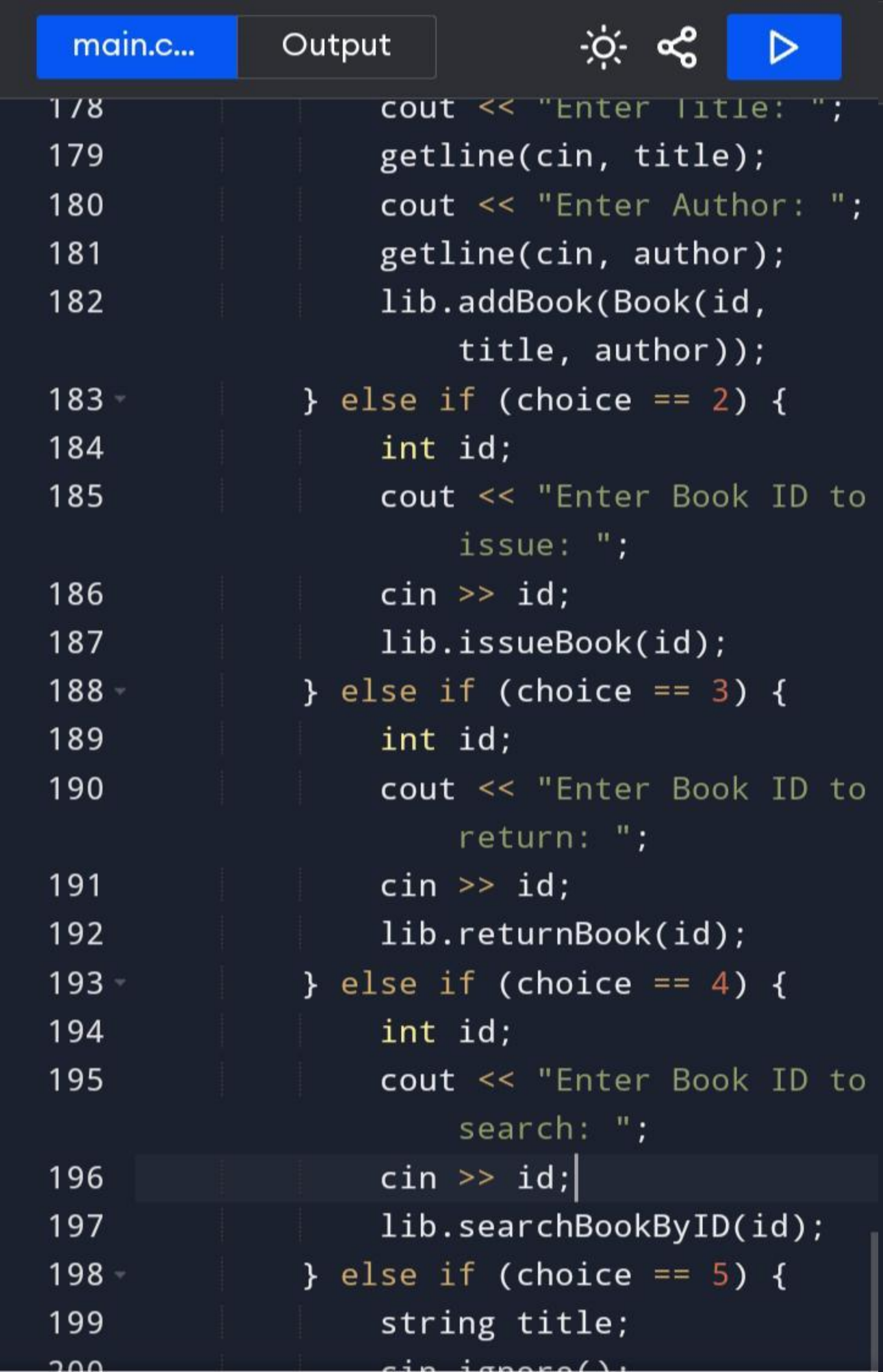
```
    }
        cout << "Book not found!\n";
    }

    void displayStats() {
        int total = books.size();
        int issued = 0, available = 0;
        for (const auto &b : books) {
            if (b.isAvailable())
                available++;
            else
                issued++;
        }
        cout << "Total Books: " << total
            << " | Available: " << available
            << " | Issued: " << issued << endl;
    }

    void saveToFile(const string &filename) {
        ofstream out(filename);
```



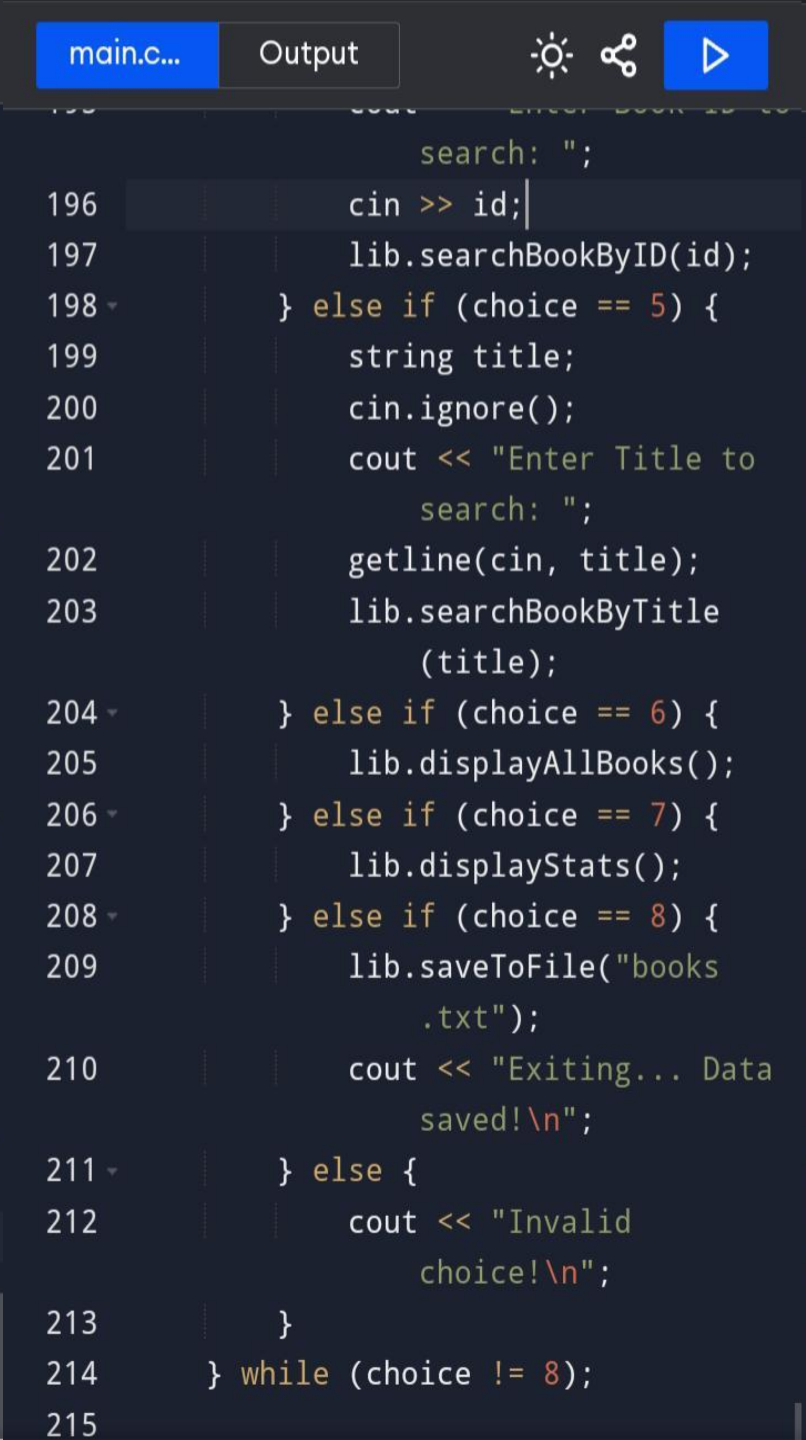
```
            b.saveToFile(out);
        }
        out.close();
    }

    void loadFromFile(const string &filename) {
        ifstream in(filename);
        string line;
        while (getline(in, line)) {
            books.push_back(Book::loadFromString(line));
        }
        in.close();
    }

    void displayAllBooks() {
        for (const auto &b : books) {
            b.display();
        }
    }
};

// Main program
```



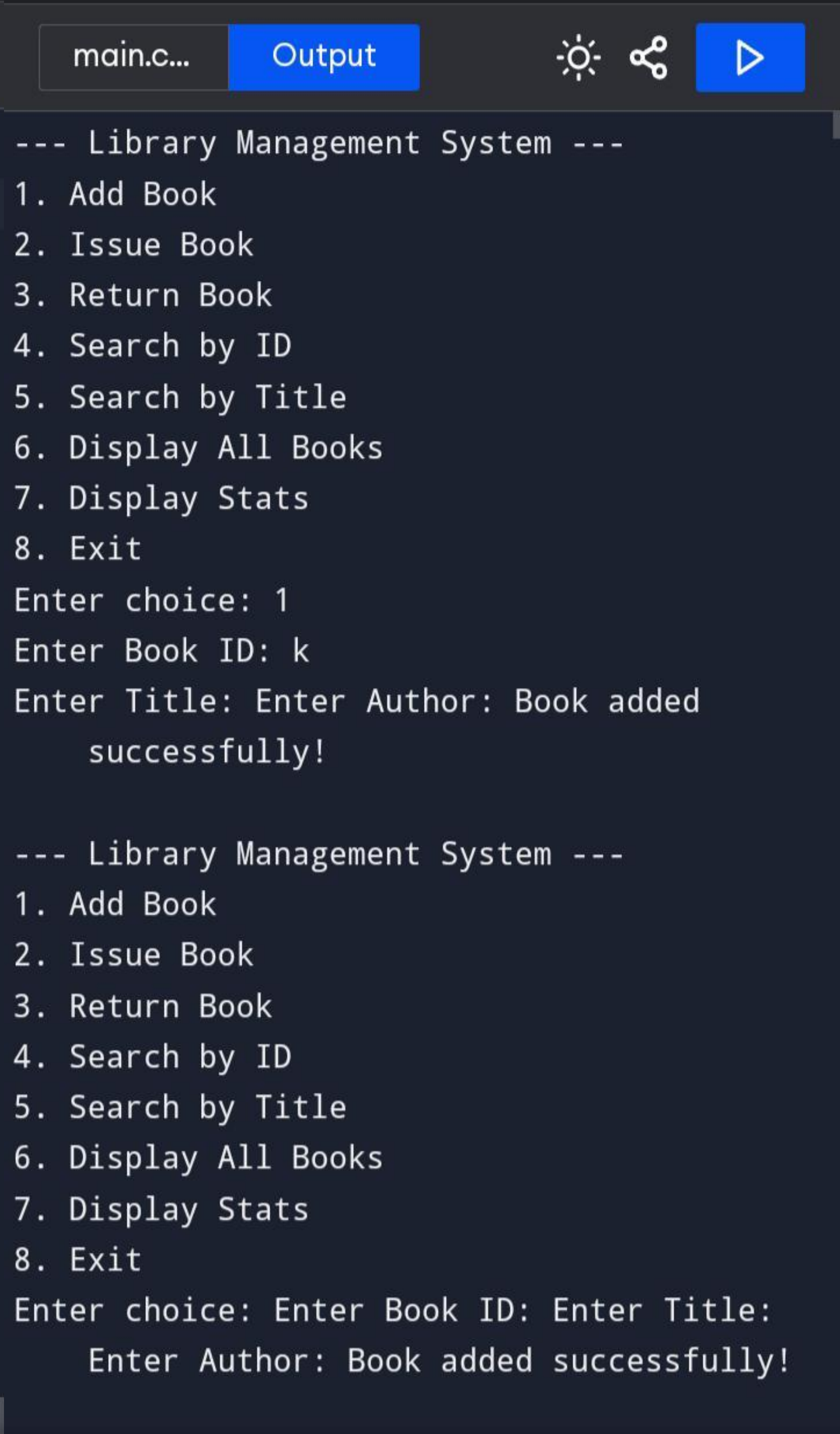
```
int main() {
    Library lib;
    lib.loadFromFile("books.txt");

    int choice;
    do {
        cout << "\n--- Library
            Management System ---\
        cout << "1. Add Book\n2.
            Issue Book\n3. Return
            Book\n4. Search by ID\
            Search by Title\n";
        cout << "6. Display All Books\n7. Display Stats\n8. Exit\n";
        cout << "Enter choice: ";
        cin >> choice;

        if (choice == 1) {
            int id;
            string title, author;
            cout << "Enter Book ID:
                ;
            cin >> id;
            cin.ignore();
            cout << "Enter Title: "
            getline(cin, title);
```



```cpp
            cout << "Enter Title: ";
            getline(cin, title);
            cout << "Enter Author: ";
            getline(cin, author);
            lib.addBook(Book(id, title, author));
        } else if (choice == 2) {
            int id;
            cout << "Enter Book ID to issue: ";
            cin >> id;
            lib.issueBook(id);
        } else if (choice == 3) {
            int id;
            cout << "Enter Book ID to return: ";
            cin >> id;
            lib.returnBook(id);
        } else if (choice == 4) {
            int id;
            cout << "Enter Book ID to search: ";
            cin >> id;
            lib.searchBookByID(id);
        } else if (choice == 5) {
            string title;
            cin.ignore();
```



```cpp
            cout << "Enter Book ID to search: ";
            cin >> id;
            lib.searchBookByID(id);
        } else if (choice == 5) {
            string title;
            cin.ignore();
            cout << "Enter Title to search: ";
            getline(cin, title);
            lib.searchBookByTitle(title);
        } else if (choice == 6) {
            lib.displayAllBooks();
        } else if (choice == 7) {
            lib.displayStats();
        } else if (choice == 8) {
            lib.saveToFile("books.txt");
            cout << "Exiting... Data saved!\n";
        } else {
            cout << "Invalid choice!\n";
        }
    } while (choice != 8);

```



```
--- Library Management System ---
1. Add Book
2. Issue Book
3. Return Book
4. Search by ID
5. Search by Title
6. Display All Books
7. Display Stats
8. Exit
Enter choice: 1
Enter Book ID: k
Enter Title: Enter Author: Book added successfully!

--- Library Management System ---
1. Add Book
2. Issue Book
3. Return Book
4. Search by ID
5. Search by Title
6. Display All Books
7. Display Stats
8. Exit
Enter choice: Enter Book ID: Enter Title: Enter Author: Book added successfully!
```

- 2. Employee Payroll System
- Scenario:
- Develop a simple payroll system for a company. The system should calculate and display the salary of employees based on their working hours and hourly rate.
- Requirements:
- Create a class Employee with attributes like name, ID, hours worked, and hourly rate.
- Implement methods to calculate the total salary and generate a salary slip.
- Provide functionality to input, update, and delete employee records.
- Include a search feature to find employees by their ID.
- Allow users to view a list of employees with their salaries and generate a summary report showing the total payroll amount.

Employee Payroll System:

- *Accurate*: The system should calculate salaries correctly based on working hours and hourly rates.

- *Efficient*: The system should quickly process employee data and generate salary slips.

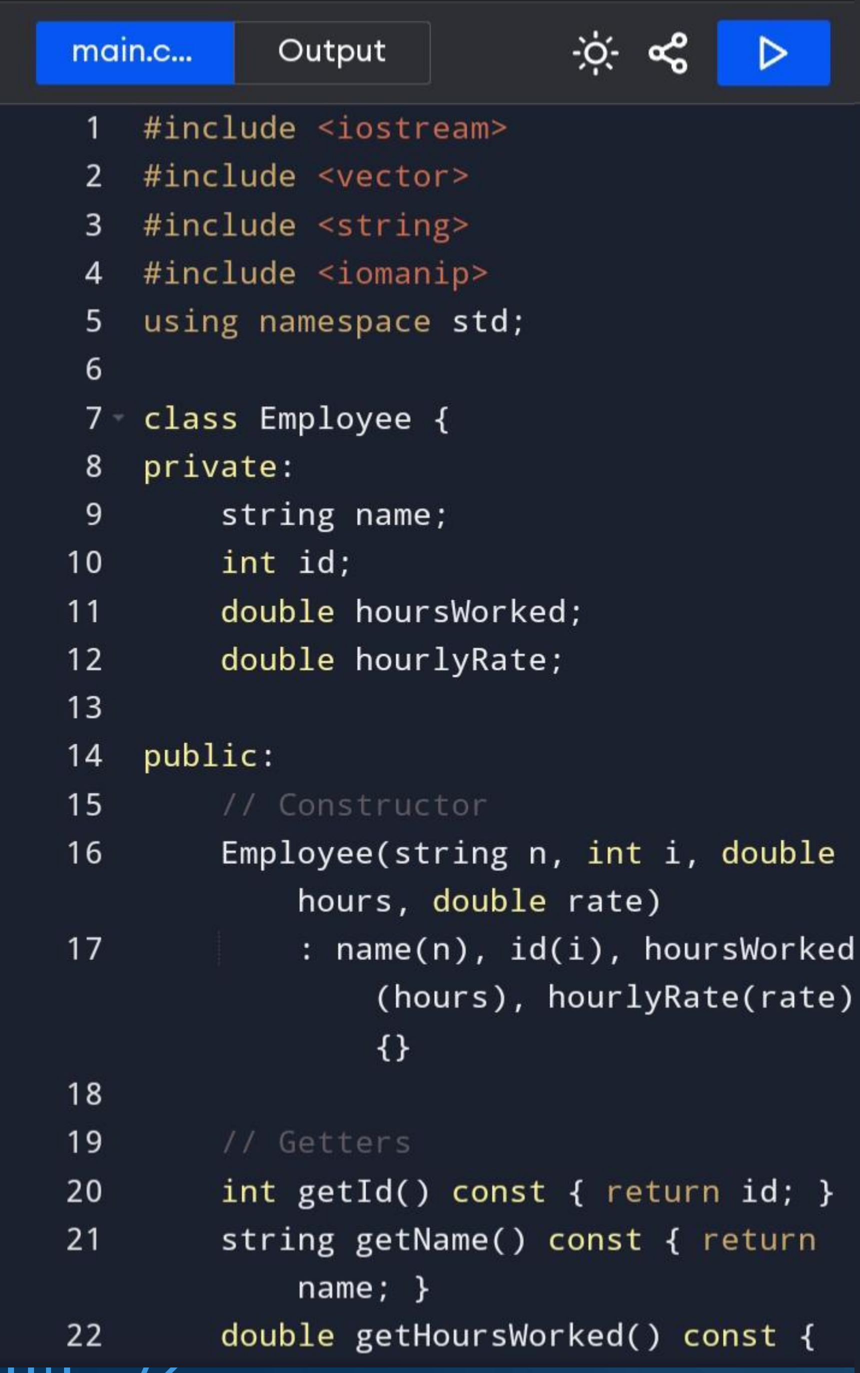
- *User-friendly*: The system should provide an intuitive interface for users to input, update, and delete employee records.

- *Informative*: The system should display detailed salary slips and provide a summary report of the total payroll amount.

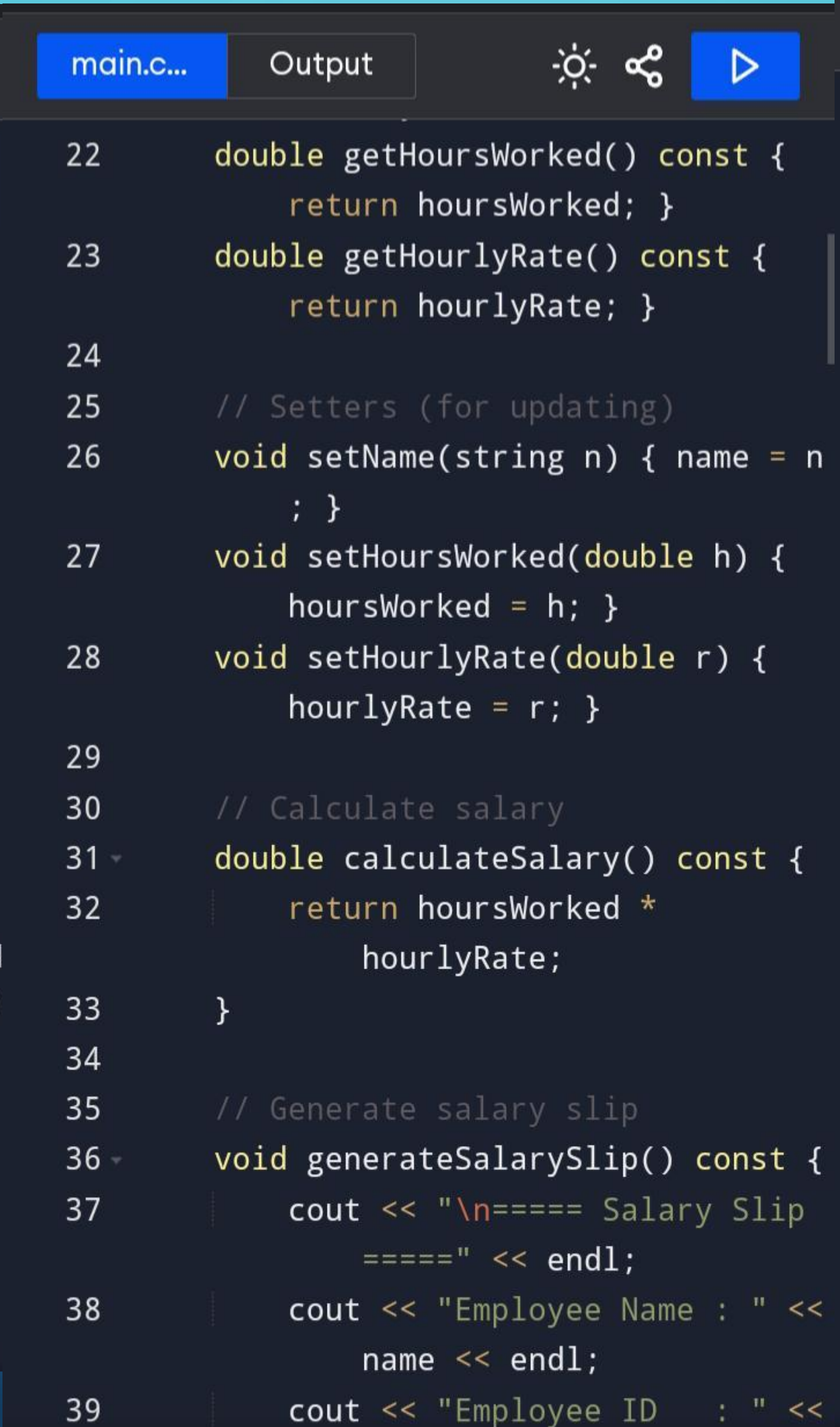
- *Flexible*: The system should allow users to easily search, update, and delete employee records.

- *Secure*: The system should protect employee data from unauthorized access or modifications.

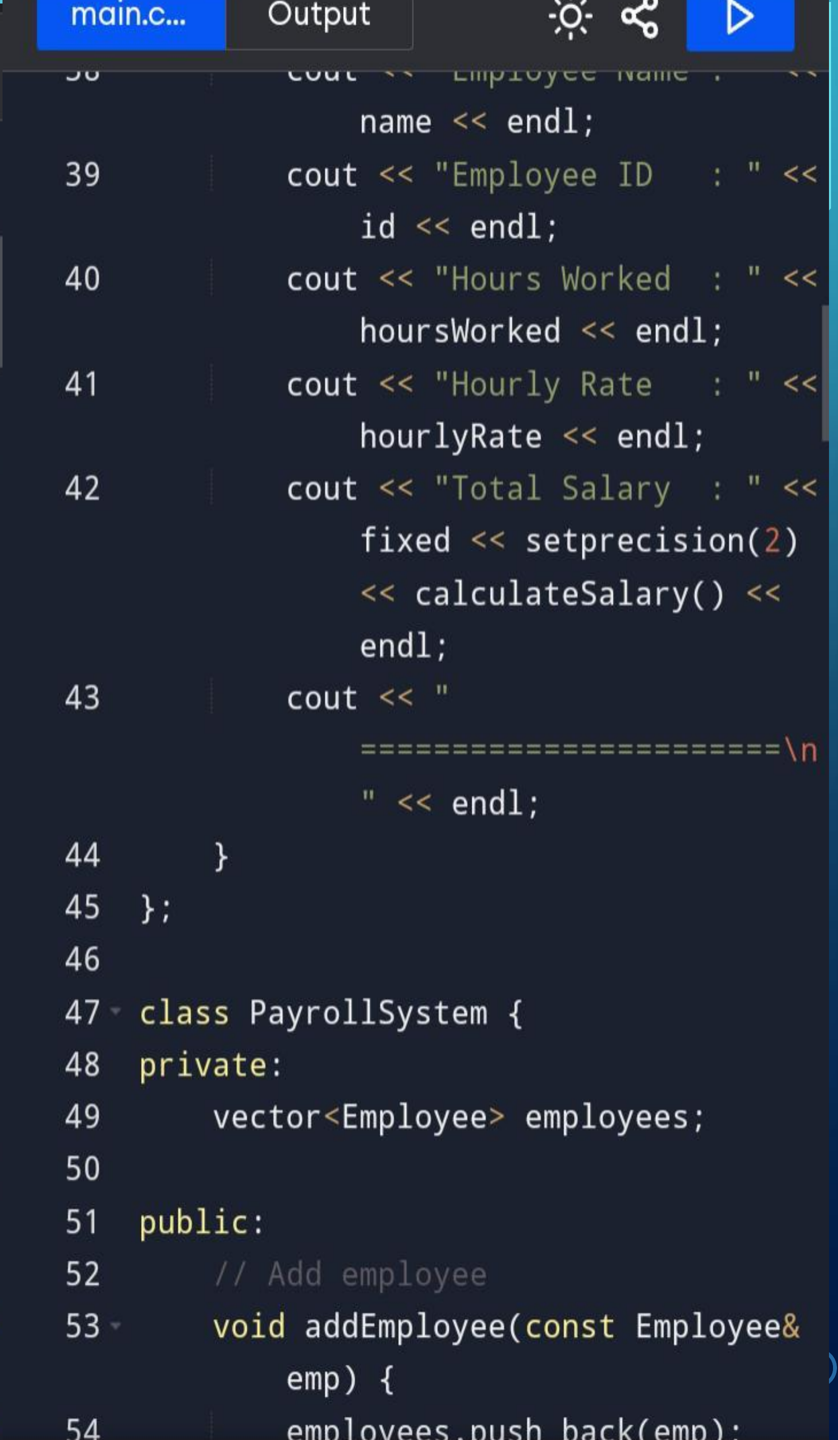
- *Scalable*: The system should be able to handle a growing number of employees and payroll calculations.



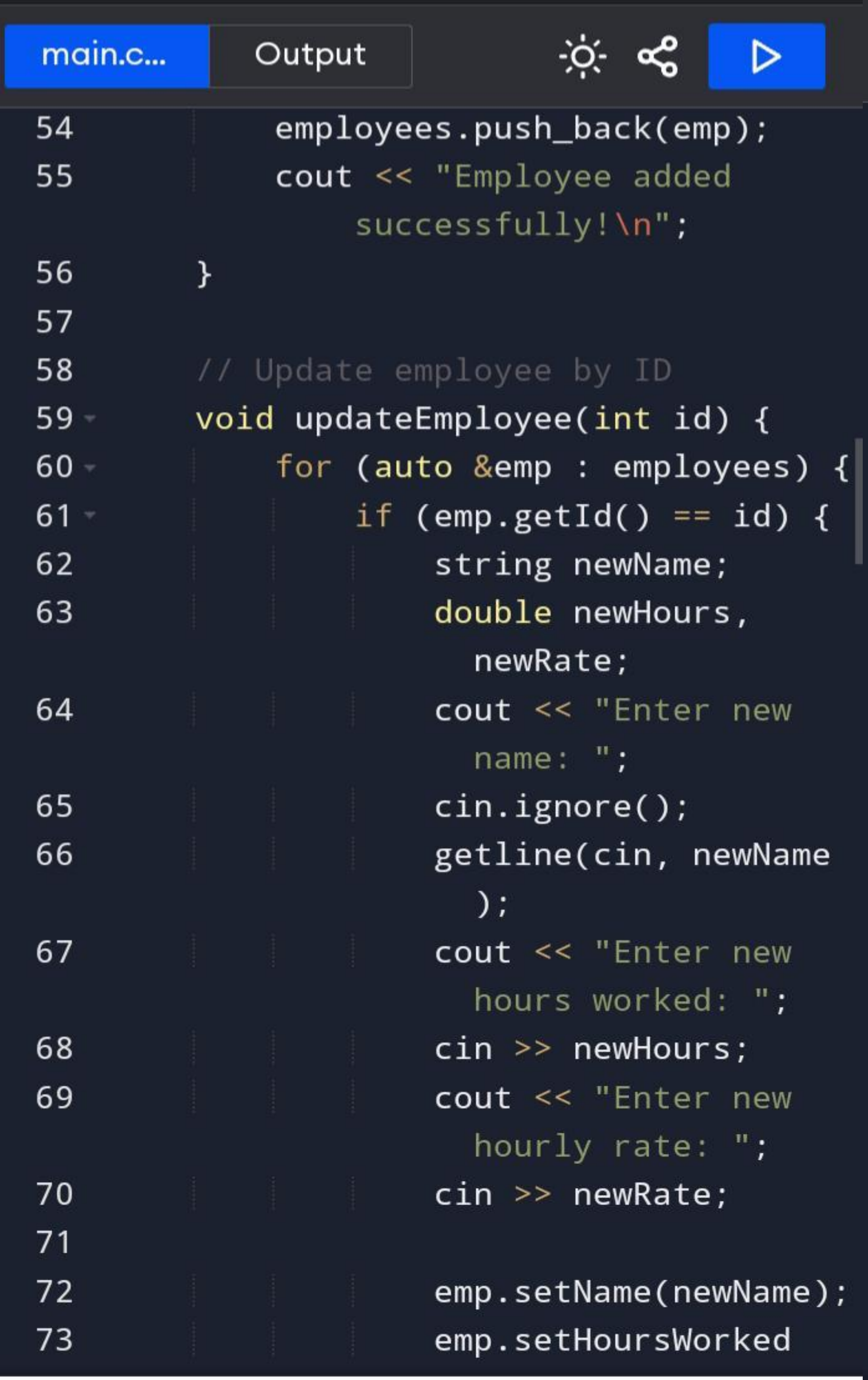
```
#include <iostream>
#include <vector>
#include <string>
#include <iomanip>
using namespace std;

class Employee {
private:
    string name;
    int id;
    double hoursWorked;
    double hourlyRate;

public:
    // Constructor
    Employee(string n, int i, double hours, double rate)
        : name(n), id(i), hoursWorked(hours), hourlyRate(rate) {}

    // Getters
    int getId() const { return id; }
    string getName() const { return name; }
    double getHoursWorked() const {
```



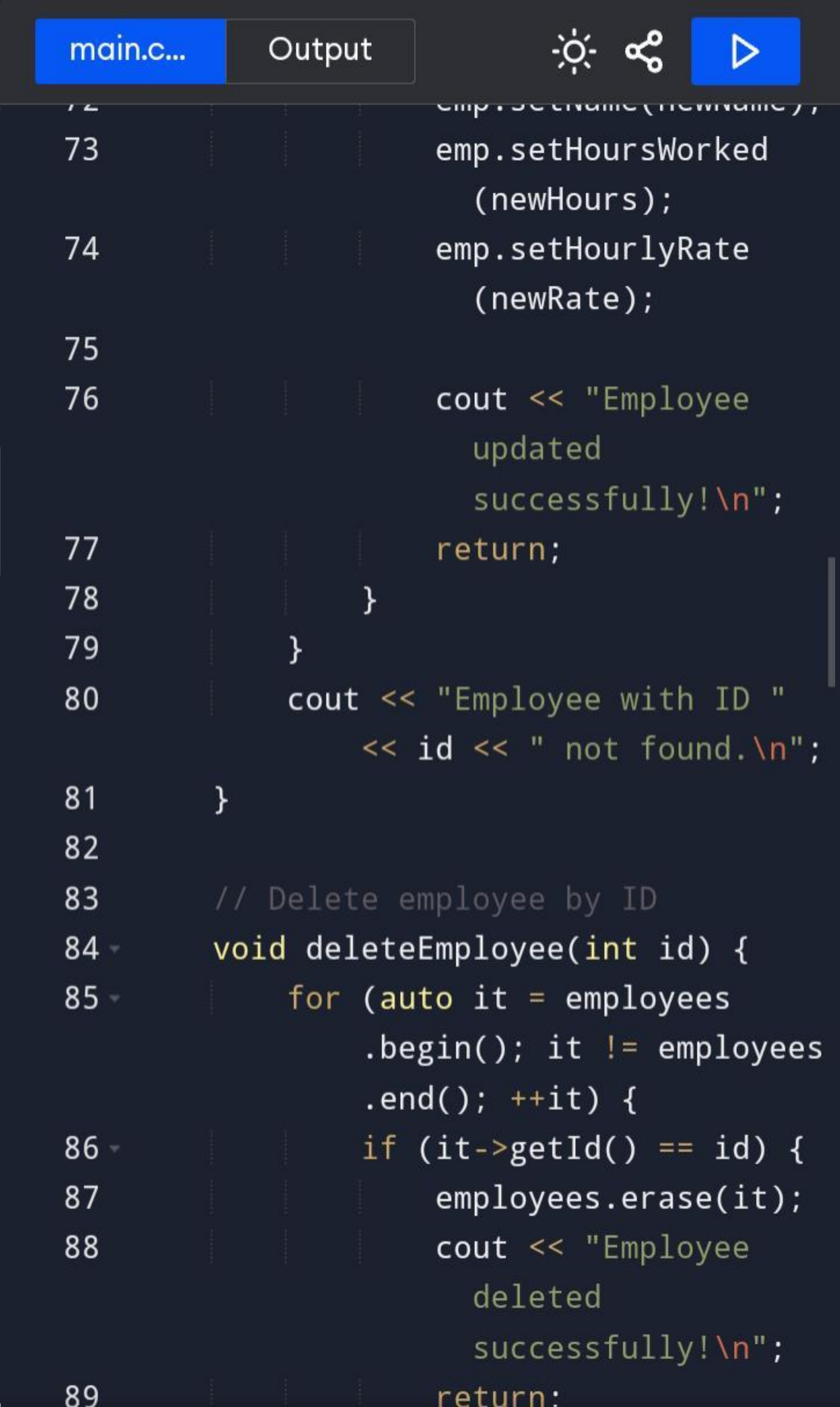
```
    double getHoursWorked() const { return hoursWorked; }
    double getHourlyRate() const { return hourlyRate; }

    // Setters (for updating)
    void setName(string n) { name = n; }
    void setHoursWorked(double h) { hoursWorked = h; }
    void setHourlyRate(double r) { hourlyRate = r; }

    // Calculate salary
    double calculateSalary() const {
        return hoursWorked * hourlyRate;
    }

    // Generate salary slip
    void generateSalarySlip() const {
        cout << "\n===== Salary Slip =====" << endl;
        cout << "Employee Name : " << name << endl;
        cout << "Employee ID   : " <<
```



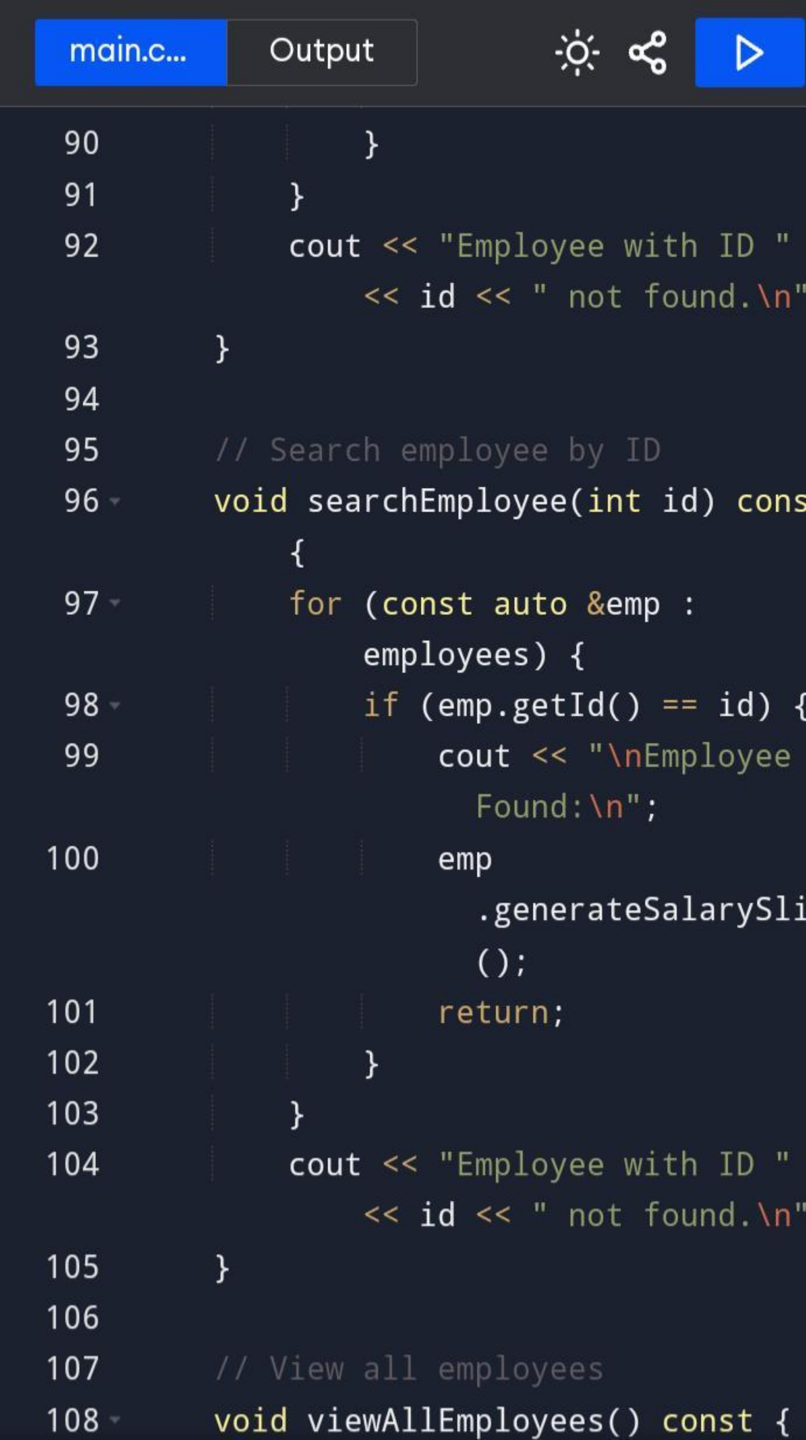
```
        cout << "Employee Name : " << name << endl;
        cout << "Employee ID   : " << id << endl;
        cout << "Hours Worked  : " << hoursWorked << endl;
        cout << "Hourly Rate   : " << hourlyRate << endl;
        cout << "Total Salary  : " << fixed << setprecision(2) << calculateSalary() << endl;
        cout << "========================\n" << endl;
    }
};

class PayrollSystem {
private:
    vector<Employee> employees;

public:
    // Add employee
    void addEmployee(const Employee& emp) {
        employees.push_back(emp);
```



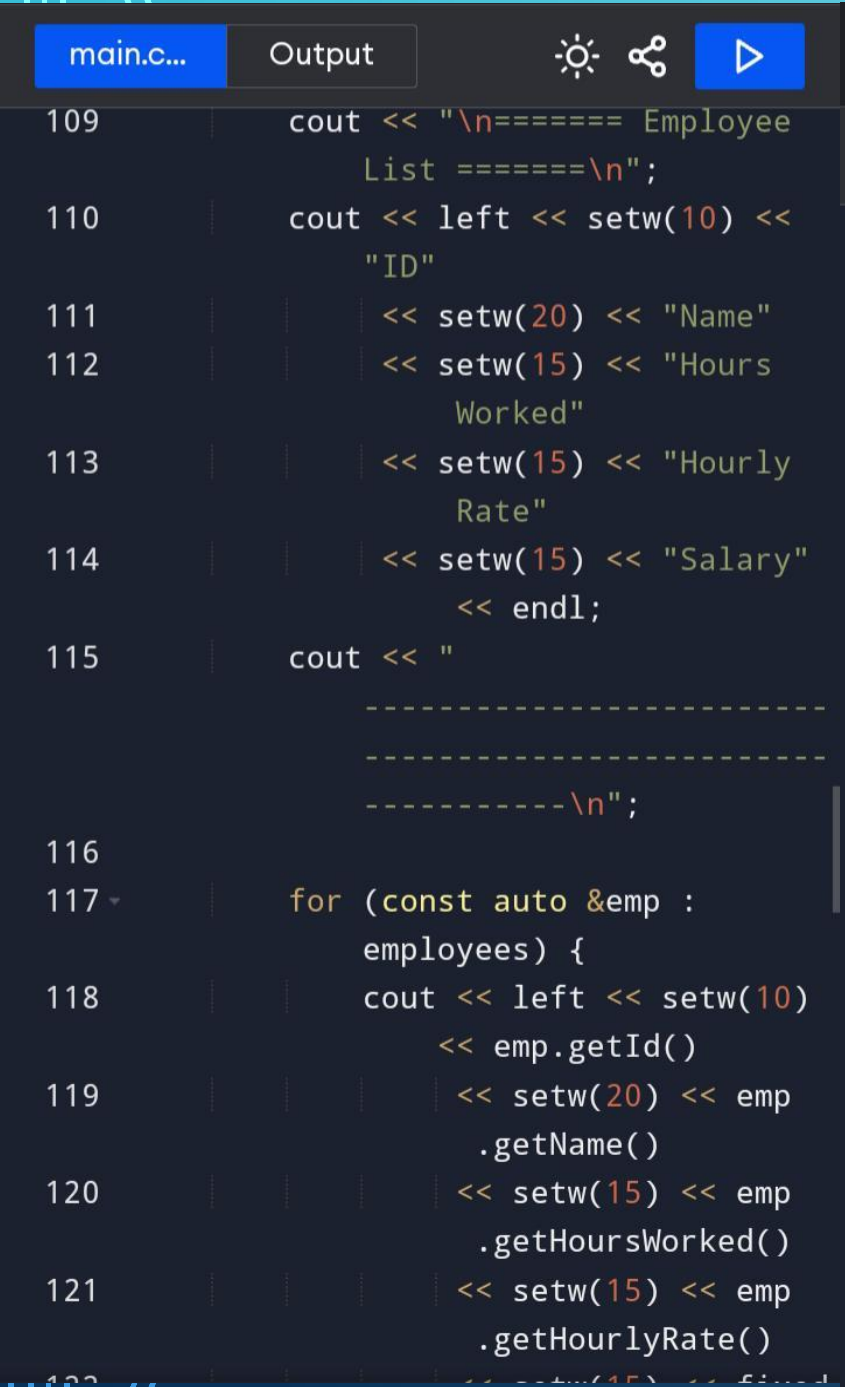
```
        employees.push_back(emp);
        cout << "Employee added successfully!\n";
    }

    // Update employee by ID
    void updateEmployee(int id) {
        for (auto &emp : employees) {
            if (emp.getId() == id) {
                string newName;
                double newHours, newRate;
                cout << "Enter new name: ";
                cin.ignore();
                getline(cin, newName);
                cout << "Enter new hours worked: ";
                cin >> newHours;
                cout << "Enter new hourly rate: ";
                cin >> newRate;

                emp.setName(newName);
                emp.setHoursWorked
```



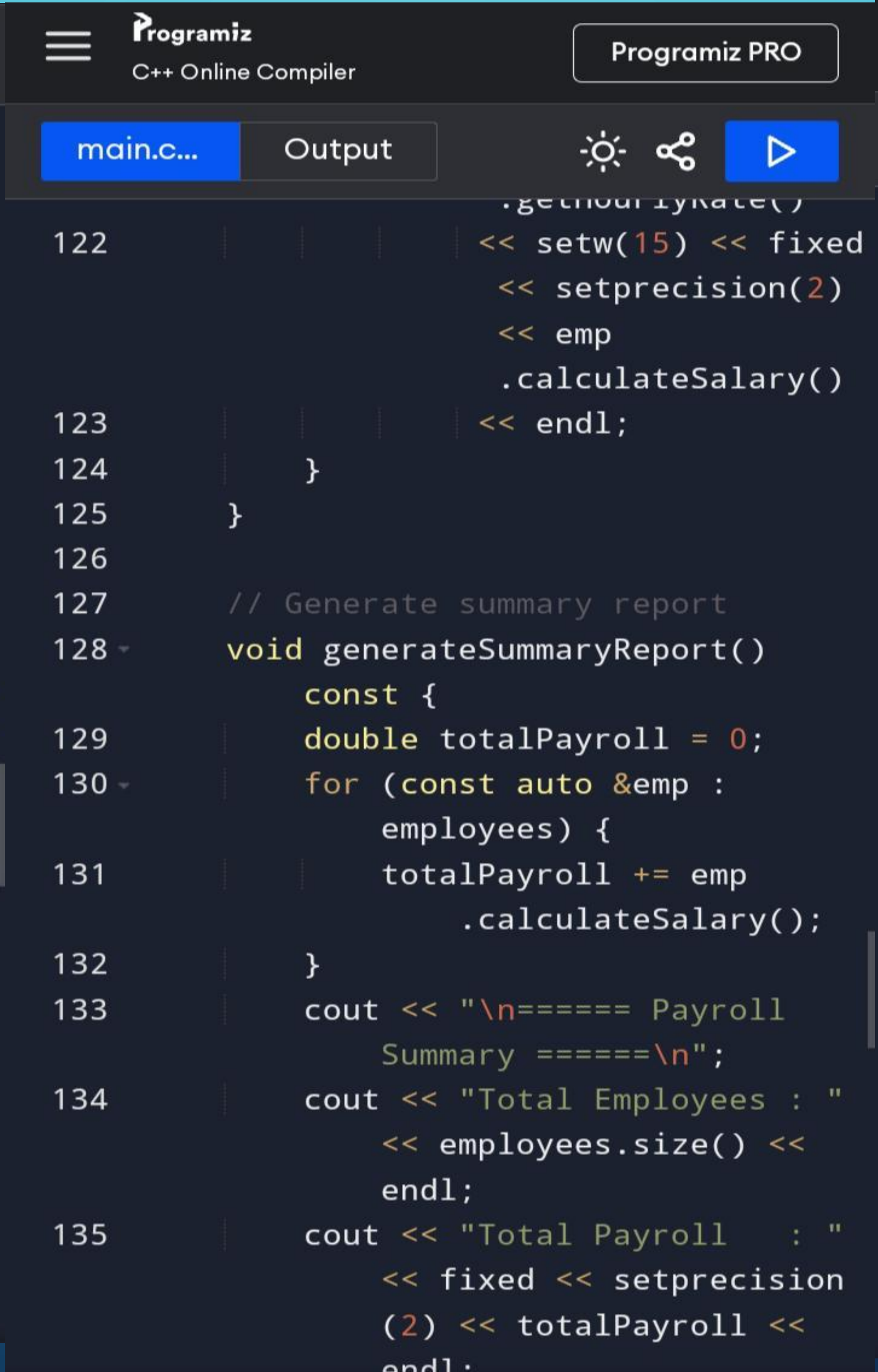
```
                emp.setName(newName);
                emp.setHoursWorked(newHours);
                emp.setHourlyRate(newRate);

                cout << "Employee updated successfully!\n";
                return;
            }
        }
        cout << "Employee with ID " << id << " not found.\n";
    }

    // Delete employee by ID
    void deleteEmployee(int id) {
        for (auto it = employees.begin(); it != employees.end(); ++it) {
            if (it->getId() == id) {
                employees.erase(it);
                cout << "Employee deleted successfully!\n";
                return;
```



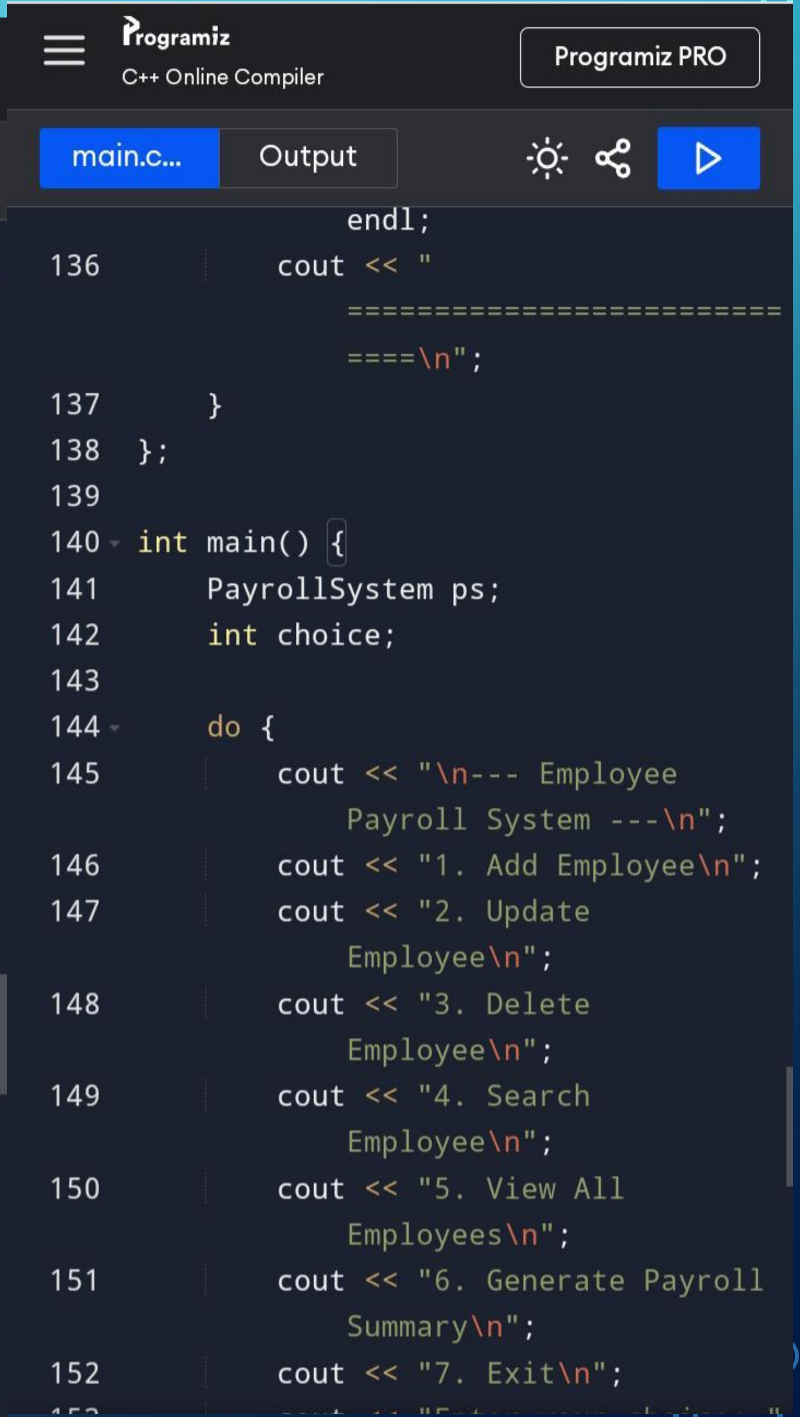
```
            }
        }
        cout << "Employee with ID " << id << " not found.\n"
    }

    // Search employee by ID
    void searchEmployee(int id) cons
    {
        for (const auto &emp : employees) {
            if (emp.getId() == id) {
                cout << "\nEmployee Found:\n";
                emp.generateSalarySli
                ();
                return;
            }
        }
        cout << "Employee with ID " << id << " not found.\n"
    }

    // View all employees
    void viewAllEmployees() const {
```



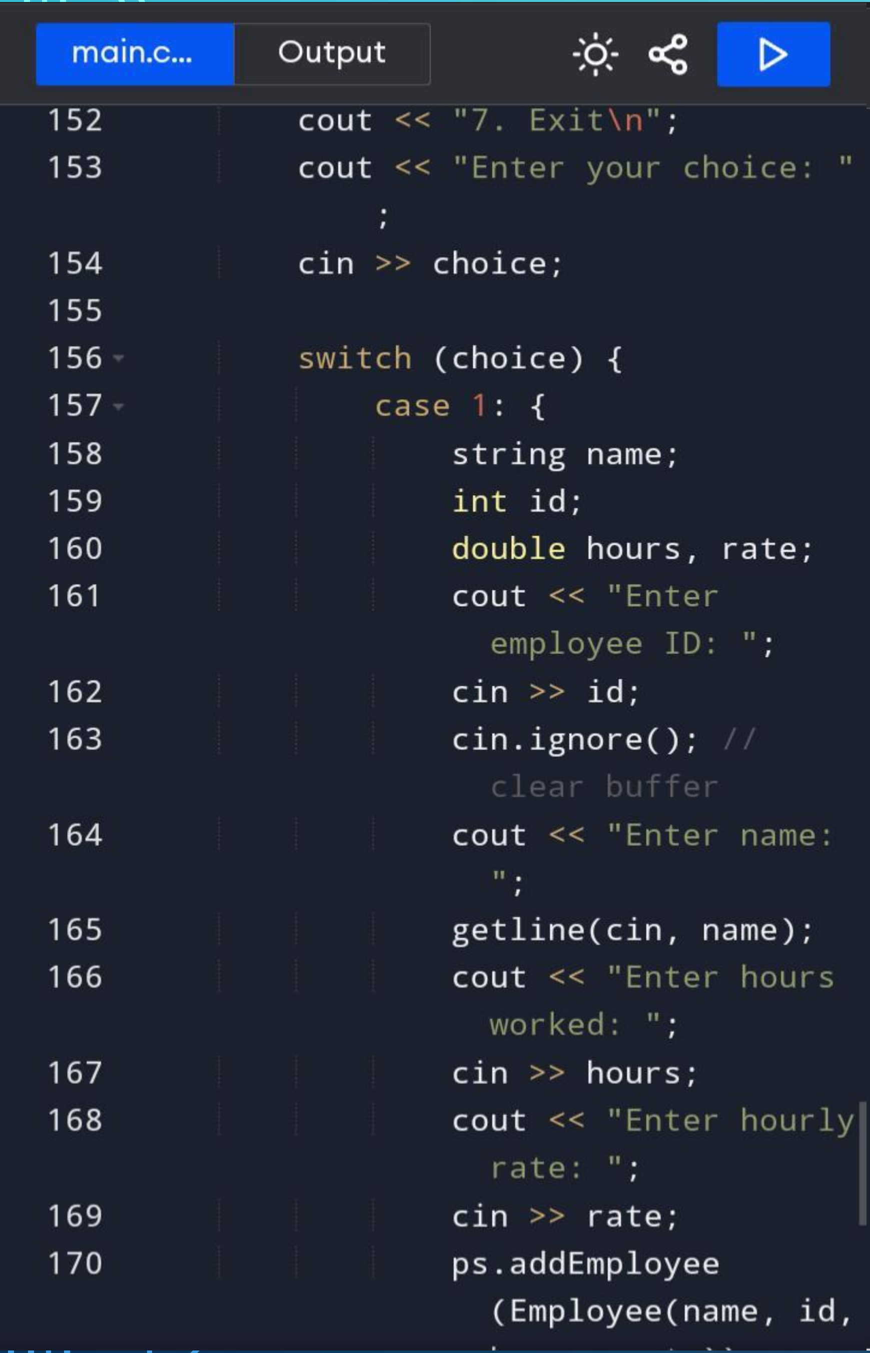
```cpp
        cout << "\n======= Employee List =======\n";
        cout << left << setw(10) << "ID"
             << setw(20) << "Name"
             << setw(15) << "Hours Worked"
             << setw(15) << "Hourly Rate"
             << setw(15) << "Salary" << endl;
        cout << "-------------------------------------------------------------\n";

        for (const auto &emp : employees) {
            cout << left << setw(10) << emp.getId()
                 << setw(20) << emp.getName()
                 << setw(15) << emp.getHoursWorked()
                 << setw(15) << emp.getHourlyRate()
                 << setw(15) << fixed
```



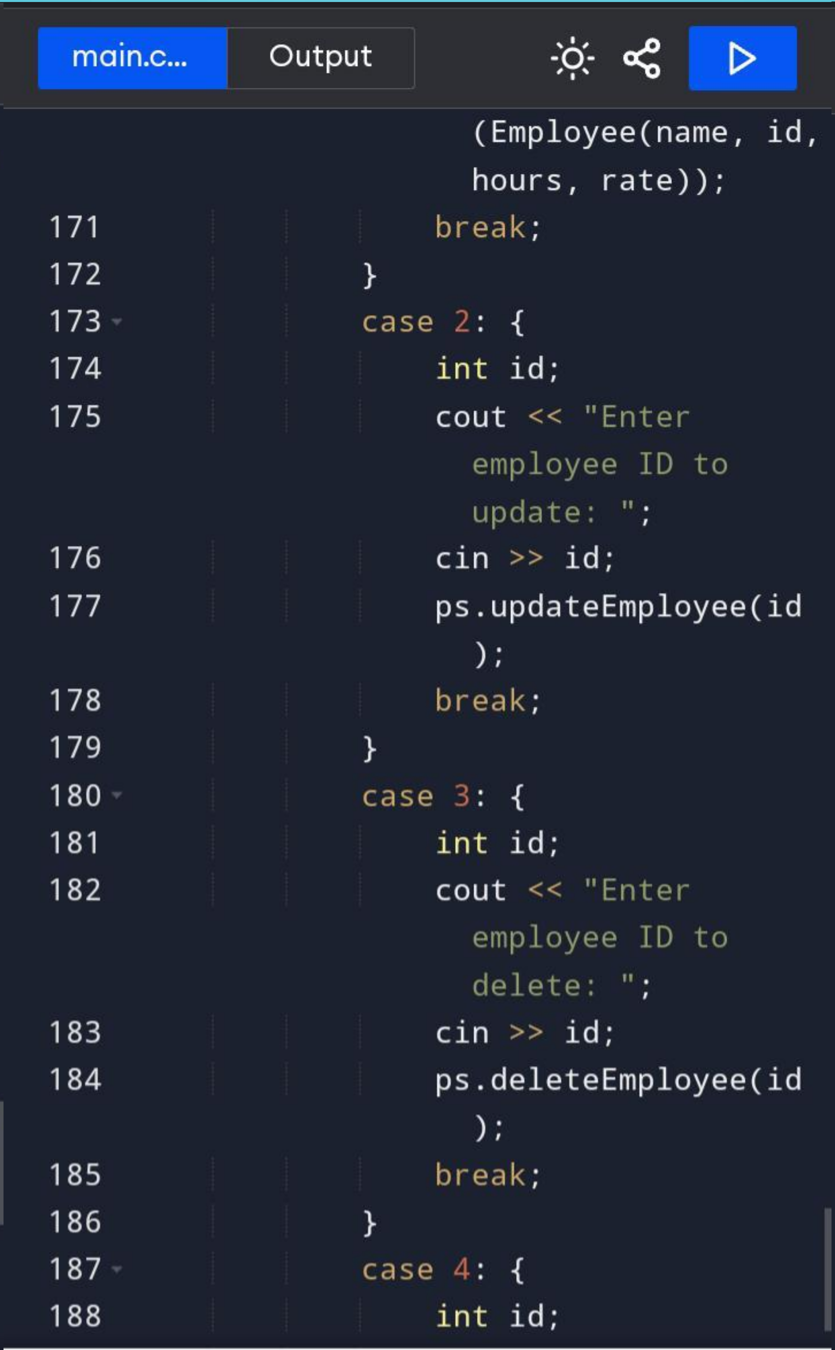
```cpp
                 .getHourlyRate()
                 << setw(15) << fixed << setprecision(2) << emp.calculateSalary()
                 << endl;
        }
    }

    // Generate summary report
    void generateSummaryReport() const {
        double totalPayroll = 0;
        for (const auto &emp : employees) {
            totalPayroll += emp.calculateSalary();
        }
        cout << "\n====== Payroll Summary ======\n";
        cout << "Total Employees : " << employees.size() << endl;
        cout << "Total Payroll   : " << fixed << setprecision(2) << totalPayroll << endl;
```



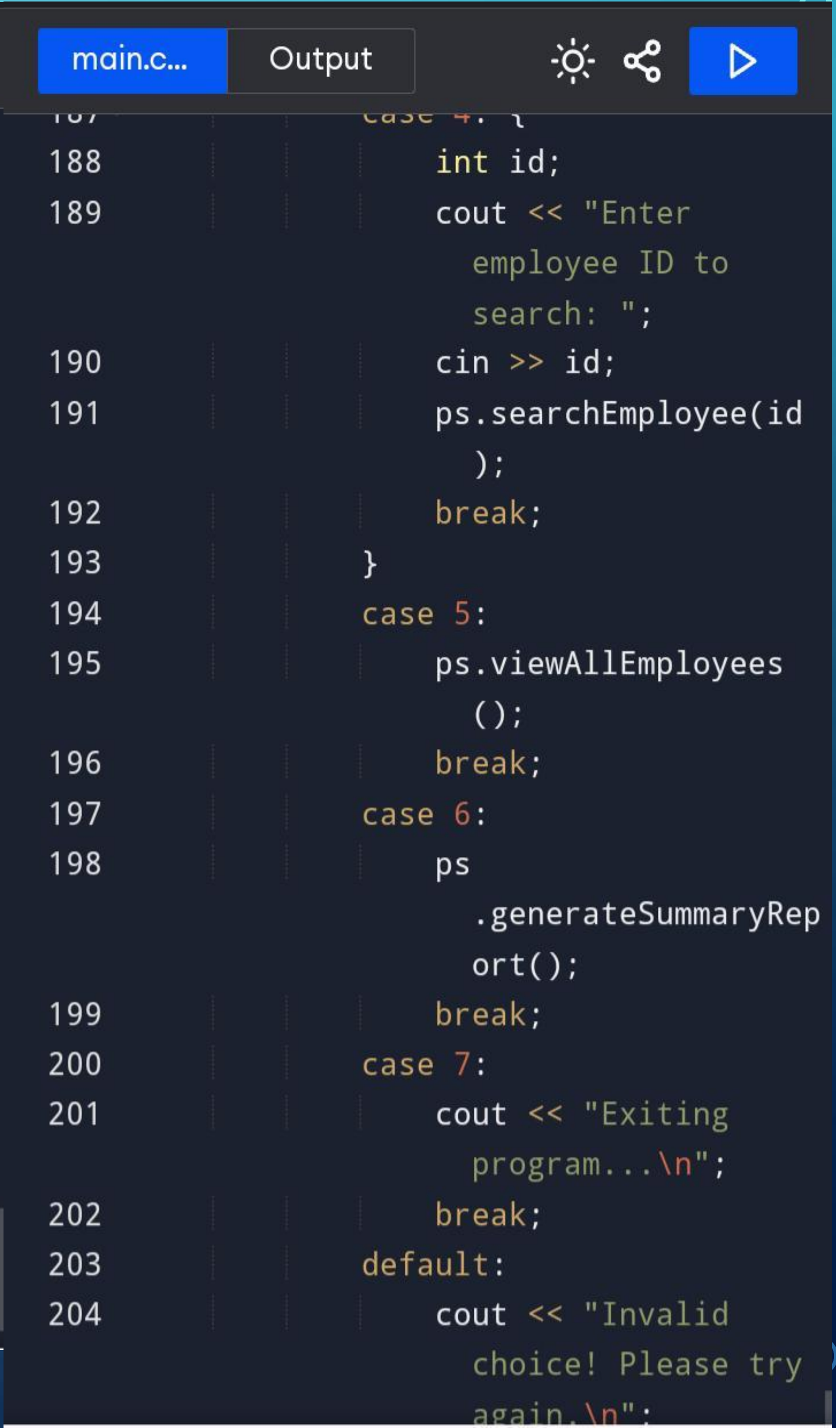
```cpp
            endl;
        cout << "=============================\n";
    }
};

int main() {
    PayrollSystem ps;
    int choice;

    do {
        cout << "\n--- Employee Payroll System ---\n";
        cout << "1. Add Employee\n";
        cout << "2. Update Employee\n";
        cout << "3. Delete Employee\n";
        cout << "4. Search Employee\n";
        cout << "5. View All Employees\n";
        cout << "6. Generate Payroll Summary\n";
        cout << "7. Exit\n";
```



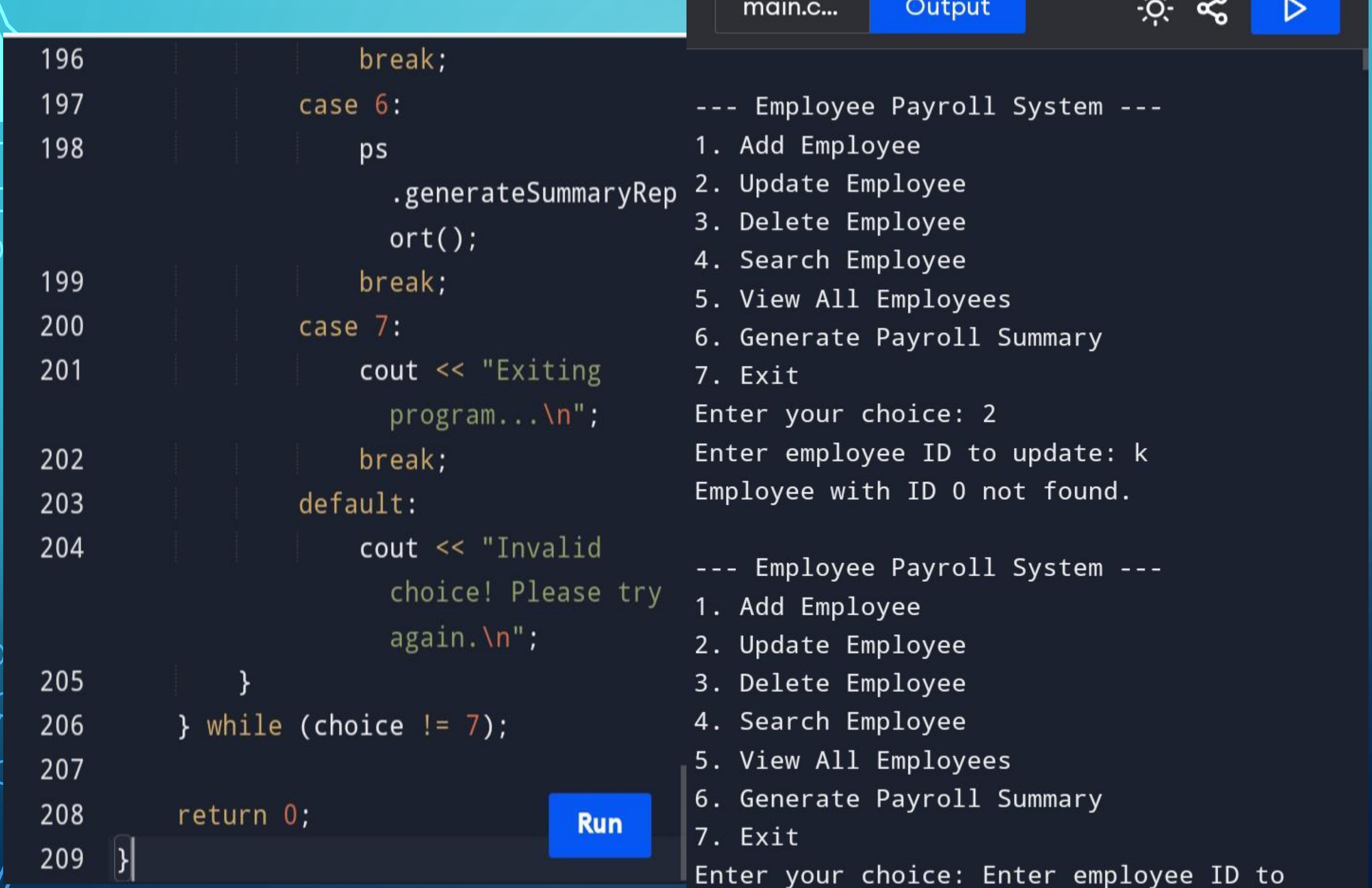
```
        cout << "7. Exit\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice) {
            case 1: {
                string name;
                int id;
                double hours, rate;
                cout << "Enter employee ID: ";
                cin >> id;
                cin.ignore(); // clear buffer
                cout << "Enter name: ";
                getline(cin, name);
                cout << "Enter hours worked: ";
                cin >> hours;
                cout << "Enter hourly rate: ";
                cin >> rate;
                ps.addEmployee(Employee(name, id,
```



```
                    (Employee(name, id, hours, rate));
                break;
            }
            case 2: {
                int id;
                cout << "Enter employee ID to update: ";
                cin >> id;
                ps.updateEmployee(id);
                break;
            }
            case 3: {
                int id;
                cout << "Enter employee ID to delete: ";
                cin >> id;
                ps.deleteEmployee(id);
                break;
            }
            case 4: {
                int id;
```



```
            case 4: {
                int id;
                cout << "Enter employee ID to search: ";
                cin >> id;
                ps.searchEmployee(id);
                break;
            }
            case 5:
                ps.viewAllEmployees();
                break;
            case 6:
                ps.generateSummaryReport();
                break;
            case 7:
                cout << "Exiting program...\n";
                break;
            default:
                cout << "Invalid choice! Please try again.\n";
```



```
                break;
            case 6:
                ps.generateSummaryReport();
                break;
            case 7:
                cout << "Exiting program...\n";
                break;
            default:
                cout << "Invalid choice! Please try again.\n";
        }
    } while (choice != 7);

    return 0;
}
```

```
--- Employee Payroll System ---
1. Add Employee
2. Update Employee
3. Delete Employee
4. Search Employee
5. View All Employees
6. Generate Payroll Summary
7. Exit
Enter your choice: 2
Enter employee ID to update: k
Employee with ID 0 not found.

--- Employee Payroll System ---
1. Add Employee
2. Update Employee
3. Delete Employee
4. Search Employee
5. View All Employees
6. Generate Payroll Summary
7. Exit
Enter your choice: Enter employee ID to
```

SUBMITTED BY
T.KARTHIKA
II BSC CS