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.

Over view:

Here's a high-level overview of a Library Management System based on your scenario and requirements:

```
#include <iostream>

#include <fstream>

#include <vector>

#include <string>

Using namespace std;


Class Book {

Private:

  Int id;

  String title;

  String author;
```

```
    Bool available;

Public:
    Book() {}
    Book(int I, string t, string a, bool avail = true)
        : id(i), title(t), author(a), available(avail) {}

    Int getId() const { return id; }
    String getTitle() const { return title; }
    String getAuthor() const { return author; }
    Bool isAvailable() const { return available; }

    Void issueBook() {
        If (available) {
            Available = false;
            Cout << “Book issued successfully!\n”;
        } else {
            Cout << “Book is already issued.\n”;
        }
    }

    Void returnBook() {
        If (!available) {
            Available = true;
            Cout << “Book returned successfully!\n”;
        } else {
```

```
        Cout << “Book was not issued.\n”;
    }
}

Void display() const {
    Cout << “ID: “ << id << “, Title: “ << title
        << “, Author: “ << author
        << “, Status: “ << (available ? “Available” : “Issued”) << endl;
}

// Save book details in file
Void saveToFile(ofstream &out) const {
    Out << id << “,” << title << “,” << author << “,” << available << “\n”;
}

// Load book details from file
Static Book loadFromString(const string &line) {
    Int id;
    String title, author;
    Bool available;
    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));
    Title = line.substr(pos1 + 1, pos2 – pos1 – 1);
```

```
    Author = line.substr(pos2 + 1, pos3 – pos2 – 1);
    Available = (line.substr(pos3 + 1) == “1”);

    Return Book(id, title, author, available);
  }
};

Class Library {
Private:
  Vector<Book> books;

Public:
  Void addBook(const Book &b) {
    Books.push_back(b);
    Cout << “Book added successfully!\n”;
  }

  Void searchBookById(int id) {
    For (auto &b : books) {
      If (b.getId() == id) {
        b.display();
        return;
      }
    }
    Cout << “Book not found.\n”;
  }
```

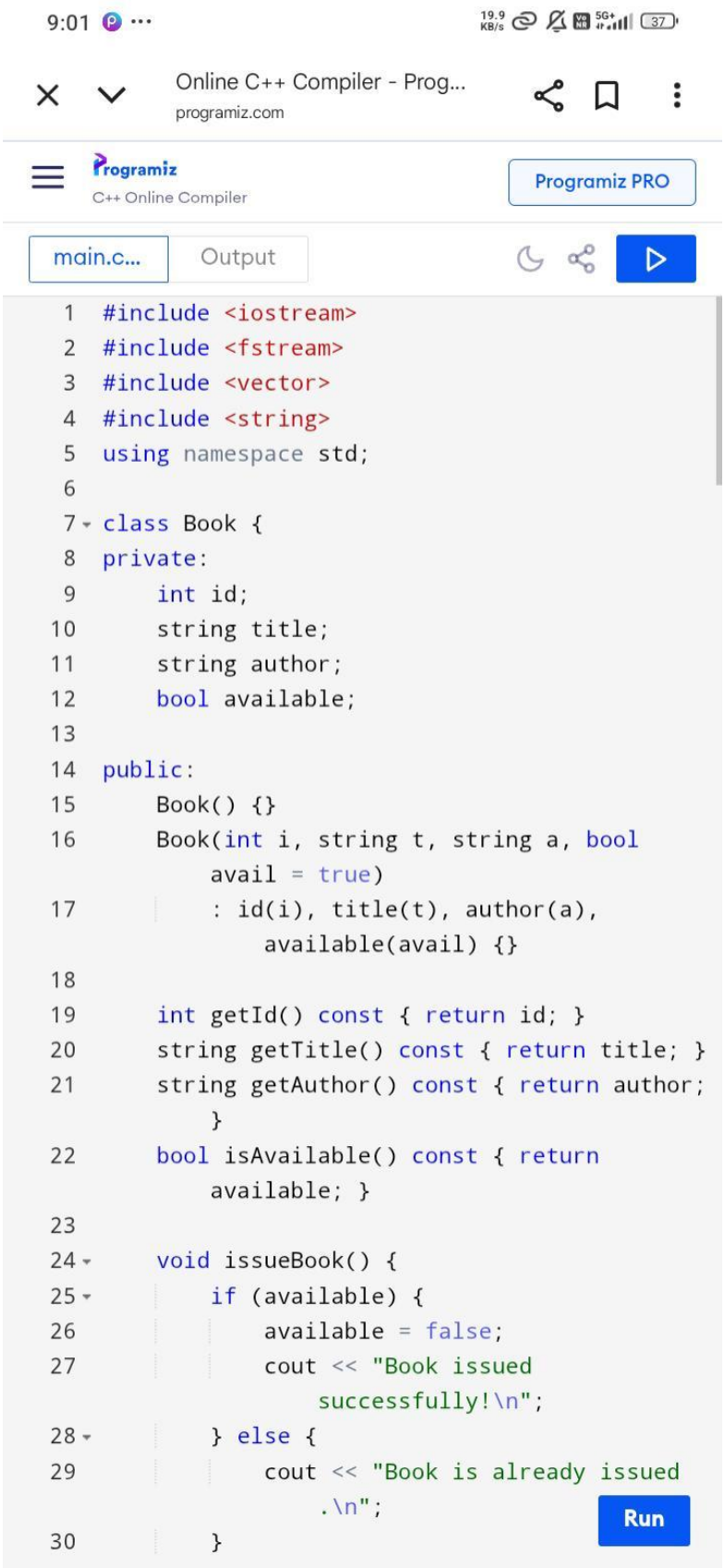
```
Void searchBookByTitle(const string &title) {
   For (auto &b : books) {
      If (b.getTitle() == title) {
         b.display();
         return;
      }
   }
   Cout << “Book not found.\n”;
}

Void issueBook(int id) {
   For (auto &b : books) {
      If (b.getId() == id) {
         b.issueBook();
         return;
      }
   }
   Cout << “Book not found.\n”;
}

Void returnBook(int id) {
   For (auto &b : books) {
      If (b.getId() == id) {
         b.returnBook();
         return;
```

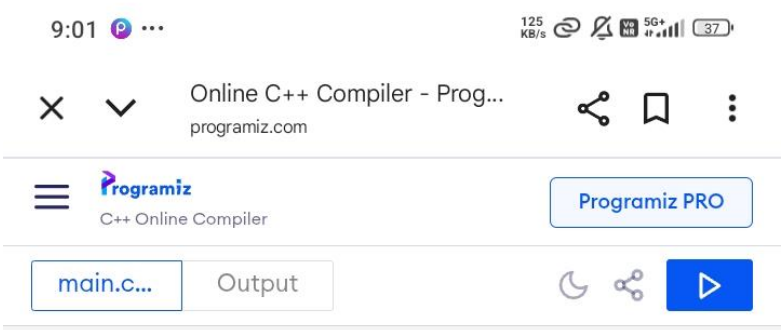
```
      }
    }
    Cout << “Book not found.\n”;
}

Void showStatistics() {
    Int total = books.size(), issued = 0, available = 0;
    For (auto &b : books) {
      If (b.isAvailable())
        Available++;
      Else
        Issued++;
    }
    Cout << “Total books: “ << total
       << “\nAvailable: “ << available
       << “\nIssued: “ << issued << endl;
}

Void saveToFile(const string &filename) {
    Ofstream out(filename);
    For (auto &b : books)
      b.saveToFile(out);
    out.close();
    cout << “Library saved to file.\n”;
}
```

```
    Void loadFromFile(const string &filename) {
        Ifstream in(filename);
        If (!in) return;

        String line;
        While (getline(in, line)) {
            If (!line.empty())
                Books.push_back(Book::loadFromString(line));
        }
        In.close();
        Cout << “Library loaded from file.\n”;
    }
};

// Driver code
Int main() {
    Library lib;
    Lib.loadFromFile(“library.txt”);

    Int choice;
    Do {
        Cout << “\n--- Library Menu ---\n”;
        Cout << “1. Add Book\n”;
        Cout << “2. Search Book by ID\n”;
        Cout << “3. Search Book by Title\n”;
        Cout << “4. Issue Book\n”;
```

```
Cout << “5. Return Book\n”;
Cout << “6. Show Statistics\n”;
Cout << “7. Save and Exit\n”;
Cout << “Enter your 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);
   Cout << “Enter Author: “;
   Getline(cin, author);
   Lib.addBook(Book(id, title, author));
} else if (choice == 2) {
   Int id;
   Cout << “Enter Book ID: “;
   Cin >> id;
   Lib.searchBookById(id);
} else if (choice == 3) {
   String title;
   Cin.ignore();
   Cout << “Enter Book Title: “;
```

```
            Getline(cin, title);
            Lib.searchBookByTitle(title);
        } else if (choice == 4) {
            Int id;
            Cout << “Enter Book ID to issue: “;
            Cin >> id;
            Lib.issueBook(id);
        } else if (choice == 5) {
            Int id;
            Cout << “Enter Book ID to return: “;
            Cin >> id;
            Lib.returnBook(id);
        } else if (choice == 6) {
            Lib.showStatistics();
        } else if (choice == 7) {
            Lib.saveToFile(“library.txt”);
            Cout << “Exiting...\n”;
        }
    } while (choice != 7);
Return 0;
}
```


9:01
Online C++ Compiler - Prog...
programiz.com
Programiz
C++ Online Compiler
Programiz PRO
main.c...
Output
#include <iostream>
#include <fstream>
#include <vector>
#include <string>
using namespace std;
class Book {
private:
int id;
string title;
string author;
bool available;
public:
Book() {}
Book(int i, string t, string a, bool avail = true)
: id(i), title(t), author(a), available(avail) {}
int getId() const { return id; }
string getTitle() const { return title; }
string getAuthor() const { return author; }
bool isAvailable() const { return available; }
void issueBook() {
if (available) {
available = false;
cout << "Book issued successfully!\n";
} else {
cout << "Book is already issued.\n";
}
Run

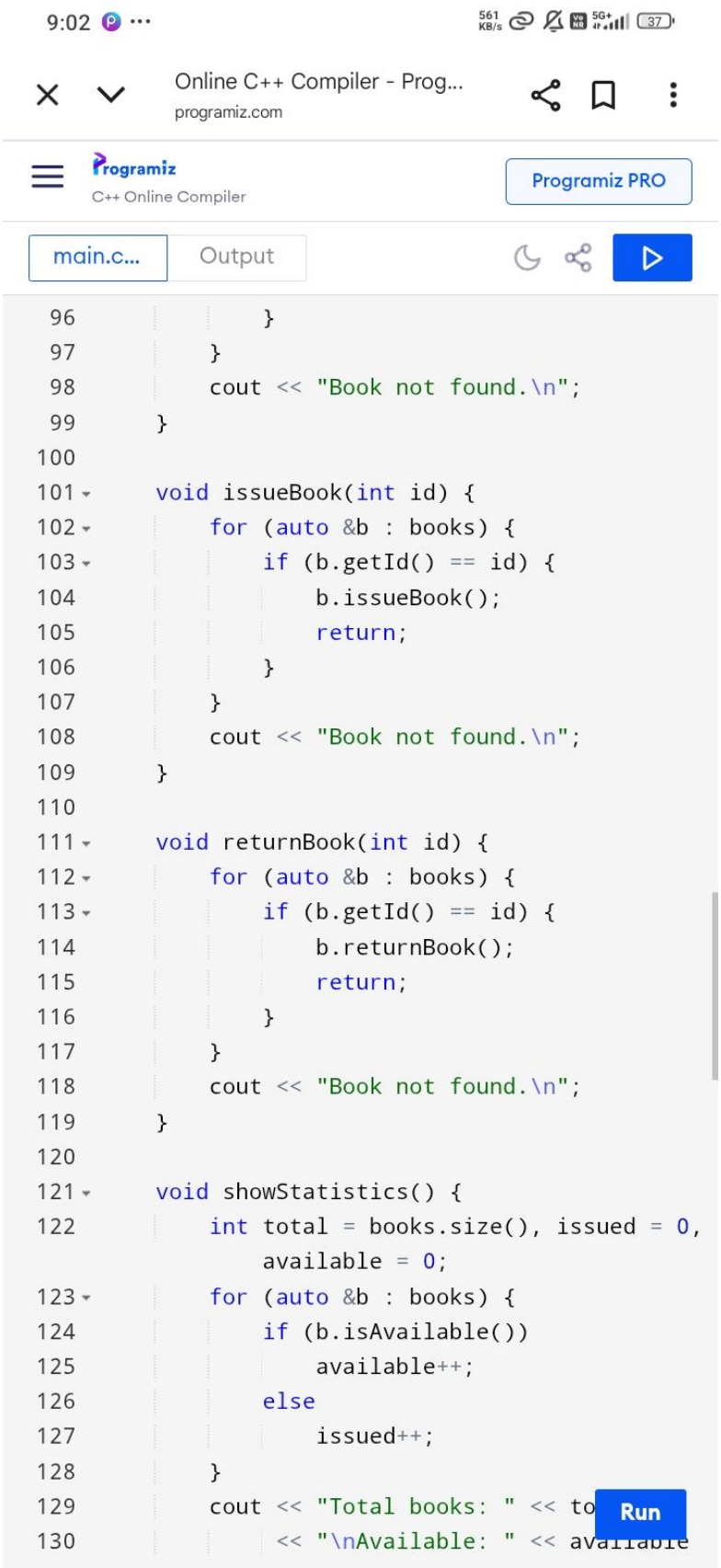



```
    }
}

void returnBook() {
    if (!available) {
        available = true;
        cout << "Book returned
            successfully!\n";
    } else {
        cout << "Book was not issued.\n";
    }
}

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

// Save book details in file
void saveToFile(ofstream &out) const {
    out << id << "," << title << "," <<
        author << "," << available <<
        "\n";
}

// Load book details from file
static Book loadFromString(const string
    &line) {
    int id;
    string title, author;
    bool available;
    size_t pos1 = line.find(",");
```

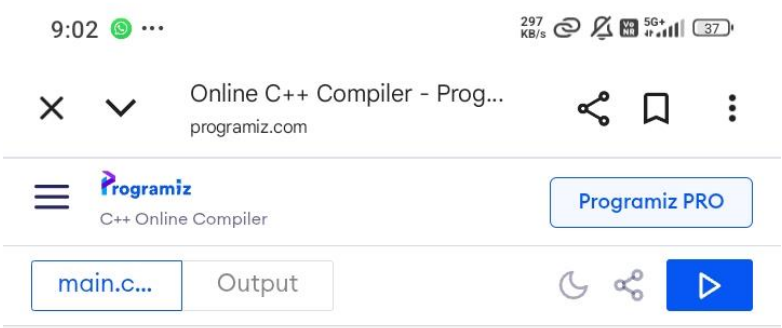
Run


```
            );
60          size_t pos3 = line.find(",", pos2 + 1
            );
61
62          id = stoi(line.substr(0, pos1));
63          title = line.substr(pos1 + 1, pos2 -
            pos1 - 1);
64          author = line.substr(pos2 + 1, pos3 -
            pos2 - 1);
65          available = (line.substr(pos3 + 1) ==
            "1");
66
67          return Book(id, title, author,
            available);
68      }
69  };
70
71  class Library {
72  private:
73      vector<Book> books;
74
75  public:
76      void addBook(const Book &b) {
77          books.push_back(b);
78          cout << "Book added successfully!\n";
79      }
80
81      void searchBookById(int id) {
82          for (auto &b : books) {
83              if (b.getId() == id) {
84                  b.display();
85                  return;
86              }
87          }
88          cout << "Book not found.\n";
89      }
```

Run

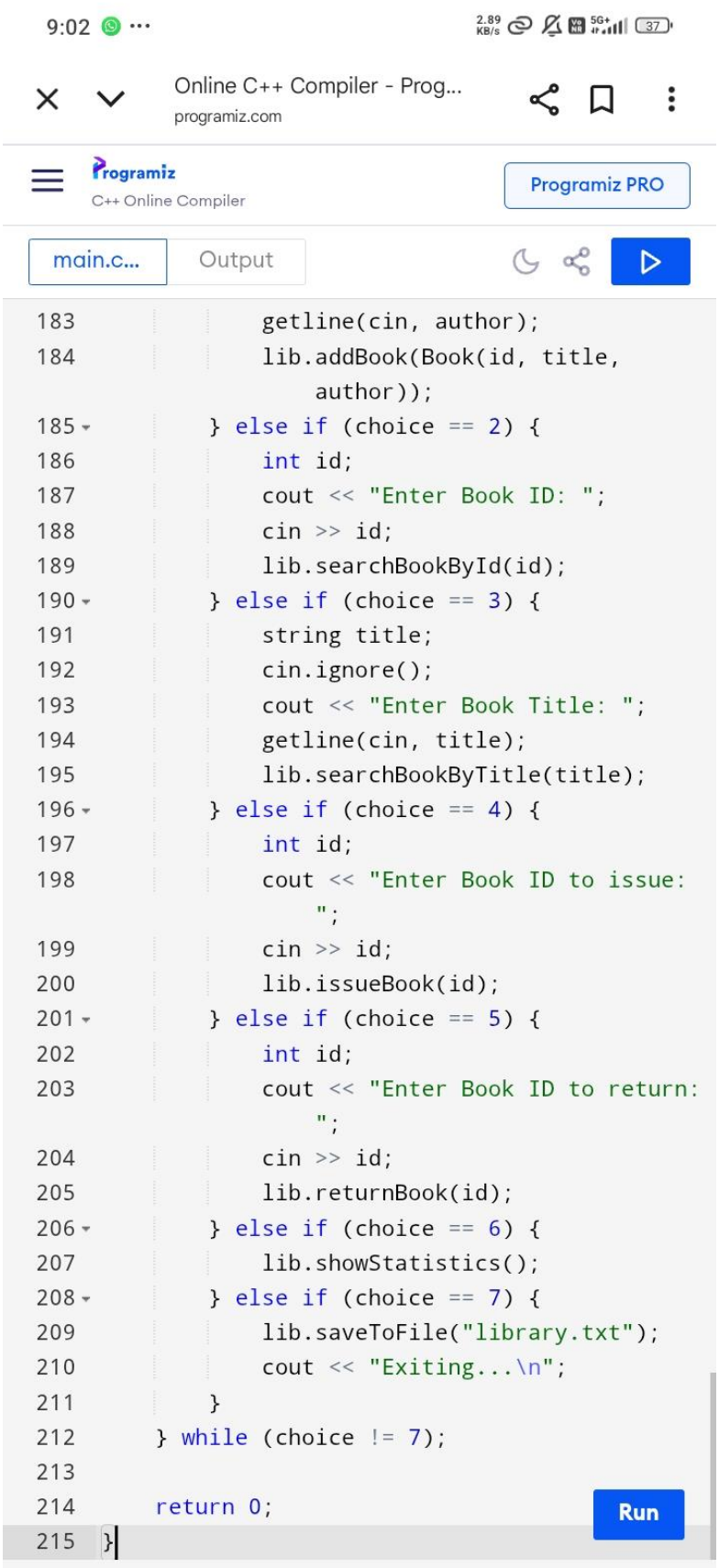



```
96          }
97      }
98      cout << "Book not found.\n";
99  }
100
101 void issueBook(int id) {
102     for (auto &b : books) {
103         if (b.getId() == id) {
104             b.issueBook();
105             return;
106         }
107     }
108     cout << "Book not found.\n";
109 }
110
111 void returnBook(int id) {
112     for (auto &b : books) {
113         if (b.getId() == id) {
114             b.returnBook();
115             return;
116         }
117     }
118     cout << "Book not found.\n";
119 }
120
121 void showStatistics() {
122     int total = books.size(), issued = 0,
        available = 0;
123     for (auto &b : books) {
124         if (b.isAvailable())
125             available++;
126         else
127             issued++;
128     }
129     cout << "Total books: " << to
130          << "\nAvailable: " << available
```



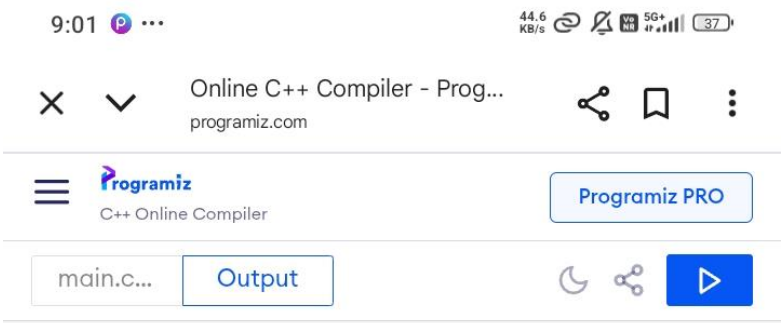

```
130             << "\nAvailable: " << available
131             << "\nIssued: " << issued <<
                    endl;
132     }
133
134     void saveToFile(const string &filename) {
135         ofstream out(filename);
136         for (auto &b : books)
137             b.saveToFile(out);
138         out.close();
139         cout << "Library saved to file.\n";
140     }
141
142     void loadFromFile(const string &filename)
            {
143         ifstream in(filename);
144         if (!in) return;
145
146         string line;
147         while (getline(in, line)) {
148             if (!line.empty())
149                 books.push_back(Book
                        ::loadFromString(line));
150         }
151         in.close();
152         cout << "Library loaded from file.\n"
                ;
153     }
154 };
155
156 // Driver code
157 int main() {
158     Library lib;
159     lib.loadFromFile("library.txt");
160
161     int choice;
```

Run




```
            getline(cin, author);
            lib.addBook(Book(id, title,
                author));
        } else if (choice == 2) {
            int id;
            cout << "Enter Book ID: ";
            cin >> id;
            lib.searchBookById(id);
        } else if (choice == 3) {
            string title;
            cin.ignore();
            cout << "Enter Book Title: ";
            getline(cin, title);
            lib.searchBookByTitle(title);
        } else if (choice == 4) {
            int id;
            cout << "Enter Book ID to issue:
                ";
            cin >> id;
            lib.issueBook(id);
        } else if (choice == 5) {
            int id;
            cout << "Enter Book ID to return:
                ";
            cin >> id;
            lib.returnBook(id);
        } else if (choice == 6) {
            lib.showStatistics();
        } else if (choice == 7) {
            lib.saveToFile("library.txt");
            cout << "Exiting...\n";
        }
    } while (choice != 7);

    return 0;
}
```


```
--- Library Menu ---
1. Add Book
2. Search Book by ID
3. Search Book by Title
4. Issue Book
5. Return Book
6. Show Statistics
7. Save and Exit
Enter your choice: 12

--- Library Menu ---
1. Add Book
2. Search Book by ID
3. Search Book by Title
4. Issue Book
5. Return Book
6. Show Statistics
7. Save and Exit
Enter your choice: 45

--- Library Menu ---
1. Add Book
2. Search Book by ID
3. Search Book by Title
4. Issue Book
5. Return Book
6. Show Statistics
7. Save and Exit
Enter your choice: 56

--- Library Menu ---
1. Add Book
2. Search Book by ID
3. Search Book by Title
4. Issue Book
```




```
Enter your choice: 12

--- Library Menu ---
1. Add Book
2. Search Book by ID
3. Search Book by Title
4. Issue Book
5. Return Book
6. Show Statistics
7. Save and Exit
Enter your choice: 45

--- Library Menu ---
1. Add Book
2. Search Book by ID
3. Search Book by Title
4. Issue Book
5. Return Book
6. Show Statistics
7. Save and Exit
Enter your choice: 56

--- Library Menu ---
1. Add Book
2. Search Book by ID
3. Search Book by Title
4. Issue Book
5. Return Book
6. Show Statistics
7. Save and Exit
Enter your choice: 7
Library saved to file.
Exiting...


=== Code Execution Successful ===
```

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 – Overview

Objective

The Employee Payroll System is designed to manage employee salary calculations in a company. It enables the organization to maintain employee records, compute salaries based on working hours and hourly rate, and generate reports for payroll management.

```
#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 h, double r) {
      Name = n;
      Id = I;
      hoursWorked = h;
      hourlyRate = r;
   }

   // Getters
   Int getID() const { return id; }
   String getName() const { return name; }
   Double getHoursWorked() const { return hoursWorked; }
   Double getHourlyRate() const { return hourlyRate; }

   // Setters (for update)
   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 generateSlip() const {
        Cout << “\n----- Salary Slip -----\n”;
        Cout << “Employee ID: “ << id << endl;
        Cout << “Name      : “ << name << endl;
        Cout << “Hours Worked: “ << hoursWorked << endl;
        Cout << “Hourly Rate : “ << hourlyRate << endl;
        Cout << “Total Salary: “ << fixed << setprecision(2) << calculateSalary() << endl;
        Cout << “----------------------\n”;
    }
};

Class PayrollSystem {
Private:
    Vector<Employee> employees;

Public:
    // Add employee
    Void addEmployee(string name, int id, double hours, double rate) {
```

```
    Employees.push_back(Employee(name, id, hours, rate));
    Cout << “Employee added successfully!\n”;
}

// Update employee
Void updateEmployee(int id, string name, double hours, double rate) {
    For (auto &emp : employees) {
        If (emp.getID() == id) {
            Emp.setName(name);
            Emp.setHoursWorked(hours);
            Emp.setHourlyRate(rate);
            Cout << “Employee updated successfully!\n”;
            Return;
        }
    }
    Cout << “Employee not found!\n”;
}

// Delete employee
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;
        }
```

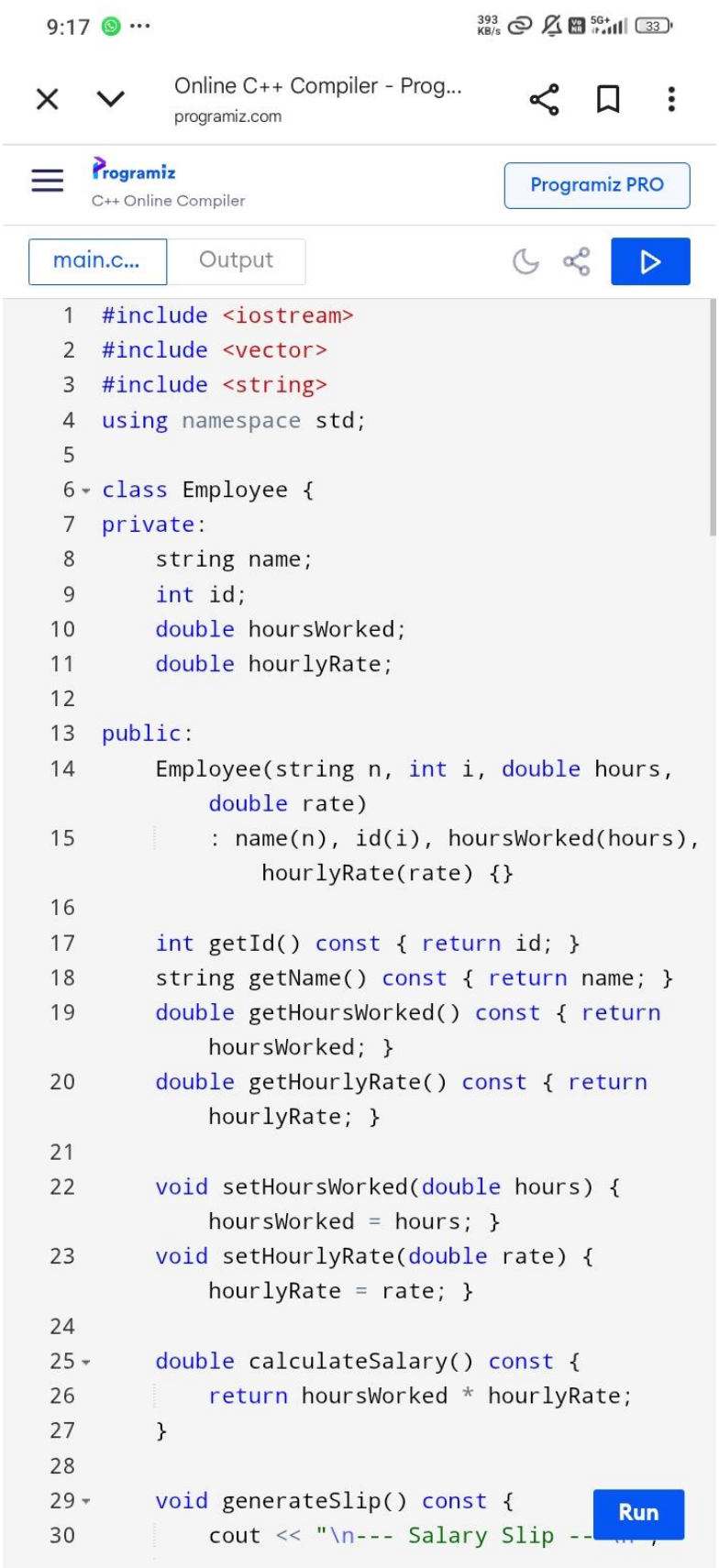
```
    }
    Cout << “Employee not found!\n”;
}

// Search employee
Void searchEmployee(int id) const {
    For (const auto &emp : employees) {
        If (emp.getID() == id) {
            Emp.generateSlip();
            Return;
        }
    }
    Cout << “Employee not found!\n”;
}

// View all employees
Void viewAllEmployees() const {
    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;

    For (const auto &emp : employees) {
        Cout << left << setw(10) << emp.getID()
```

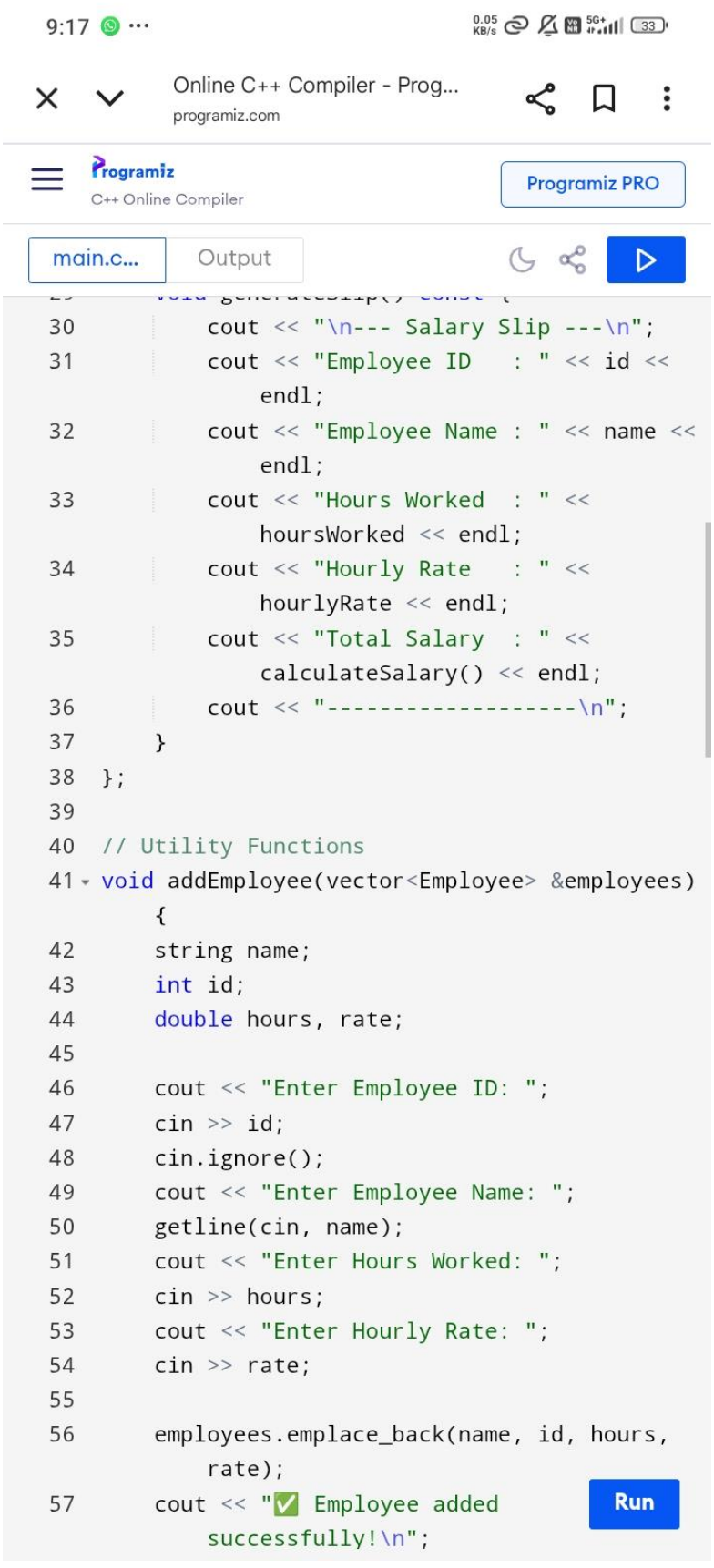
```
            << setw(20) << emp.getName()
            << setw(15) << emp.getHoursWorked()
            << setw(15) << emp.getHourlyRate()
            << setw(15) << emp.calculateSalary() << endl;
        }
    }

    // Generate summary report
    Void generateSummaryReport() const {
        Double totalPayroll = 0;
        For (const auto &emp : employees) {
            totalPayroll += emp.calculateSalary();
        }
        Cout << “\n--- Payroll Summary Report ---\n”;
        Cout << “Total Employees: “ << employees.size() << endl;
        Cout << “Total Payroll Amount: “ << fixed << setprecision(2) << totalPayroll << endl;
    }
};

Int main() {
    PayrollSystem system;
    Int choice, id;
    String name;
    Double hours, rate;

    Do {
```

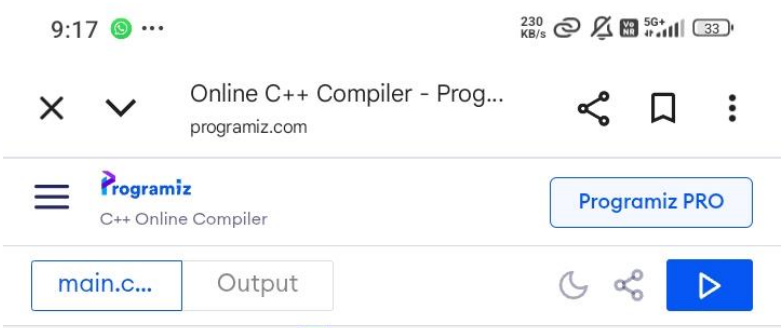
```
Cout << “\n--- Employee Payroll System ---\n”;
Cout << “1. Add Employee\n”;
Cout << “2. Update Employee\n”;
Cout << “3. Delete Employee\n”;
Cout << “4. Search Employee\n”;
Cout << “5. View All Employees\n”;
Cout << “6. Generate Summary Report\n”;
Cout << “7. Exit\n”;
Cout << “Enter your choice: “;
Cin >> choice;

Switch (choice) {
  Case 1:
    Cout << “Enter Name: “;
    Cin.ignore();
    Getline(cin, name);
    Cout << “Enter ID: “;
    Cin >> id;
    Cout << “Enter Hours Worked: “;
    Cin >> hours;
    Cout << “Enter Hourly Rate: “;
    Cin >> rate;
    System.addEmployee(name, id, hours, rate);
    Break;

  Case 2:
```

```
    Cout << “Enter ID of Employee to Update: “;
    Cin >> id;
    Cout << “Enter New Name: “;
    Cin.ignore();
    Getline(cin, name);
    Cout << “Enter New Hours Worked: “;
    Cin >> hours;
    Cout << “Enter New Hourly Rate: “;
    Cin >> rate;
    System.updateEmployee(id, name, hours, rate);
    Break;

Case 3:
    Cout << “Enter ID of Employee to Delete: “;
    Cin >> id;
    System.deleteEmployee(id);
    Break;

Case 4:
    Cout << “Enter ID of Employee to Search: “;
    Cin >> id;
    System.searchEmployee(id);
    Break;

Case 5:
    System.viewAllEmployees();
```

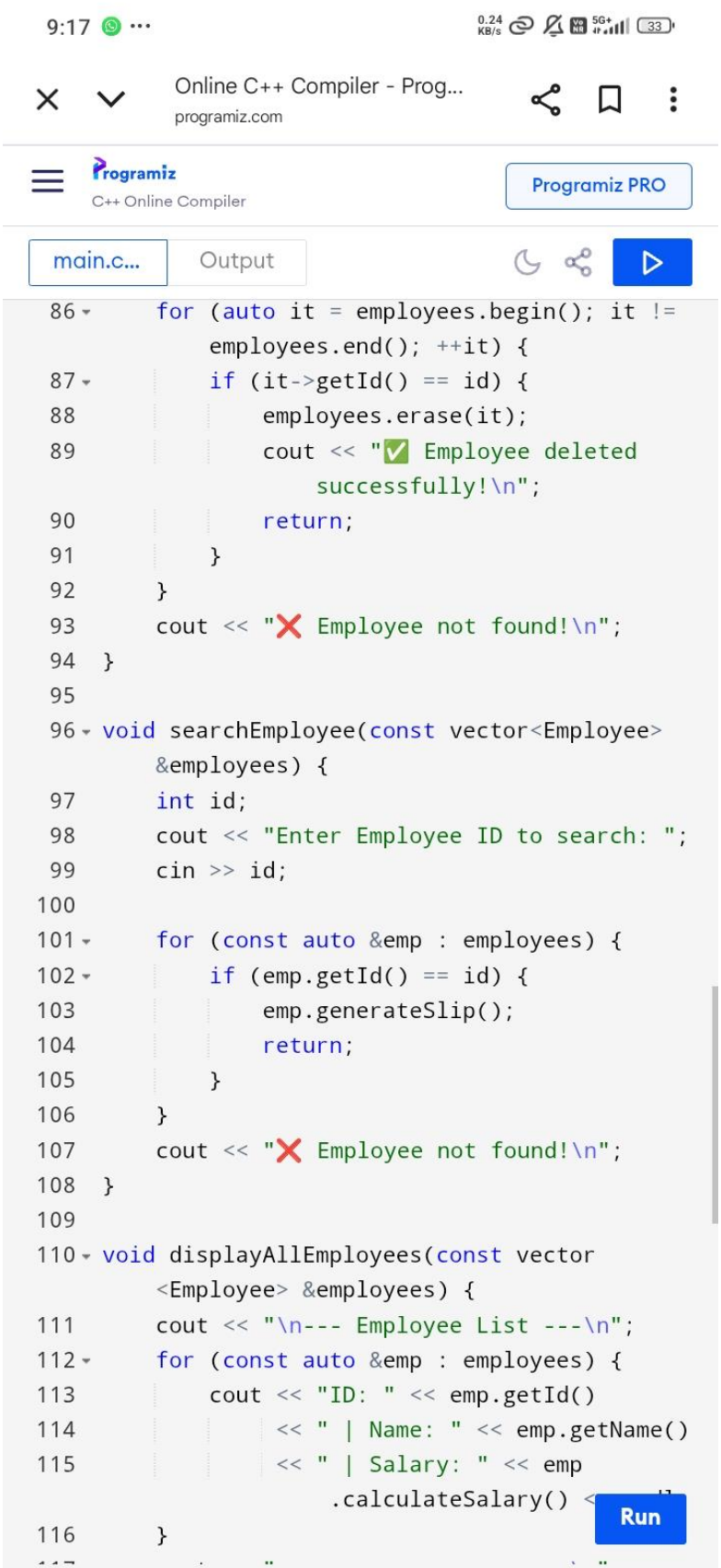
```
            Break;

        Case 6:
            System.generateSummaryReport();
            Break;

        Case 7:
            Cout << “Exiting...\n”;
            Break;

        Default:
            Cout << “Invalid choice!\n”;
      }
   } while (choice != 7);

   Return 0;
}
```


9:17
Online C++ Compiler - Prog...
programiz.com
Programiz
C++ Online Compiler
Programiz PRO
main.c...
Output
#include <iostream>
#include <vector>
#include <string>
using namespace std;
class Employee {
private:
string name;
int id;
double hoursWorked;
double hourlyRate;
public:
Employee(string n, int i, double hours,
double rate)
: name(n), id(i), hoursWorked(hours),
hourlyRate(rate) {}
int getId() const { return id; }
string getName() const { return name; }
double getHoursWorked() const { return
hoursWorked; }
double getHourlyRate() const { return
hourlyRate; }
void setHoursWorked(double hours) {
hoursWorked = hours; }
void setHourlyRate(double rate) {
hourlyRate = rate; }
double calculateSalary() const {
return hoursWorked * hourlyRate;
}
void generateSlip() const {
cout << "\n--- Salary Slip --
Run

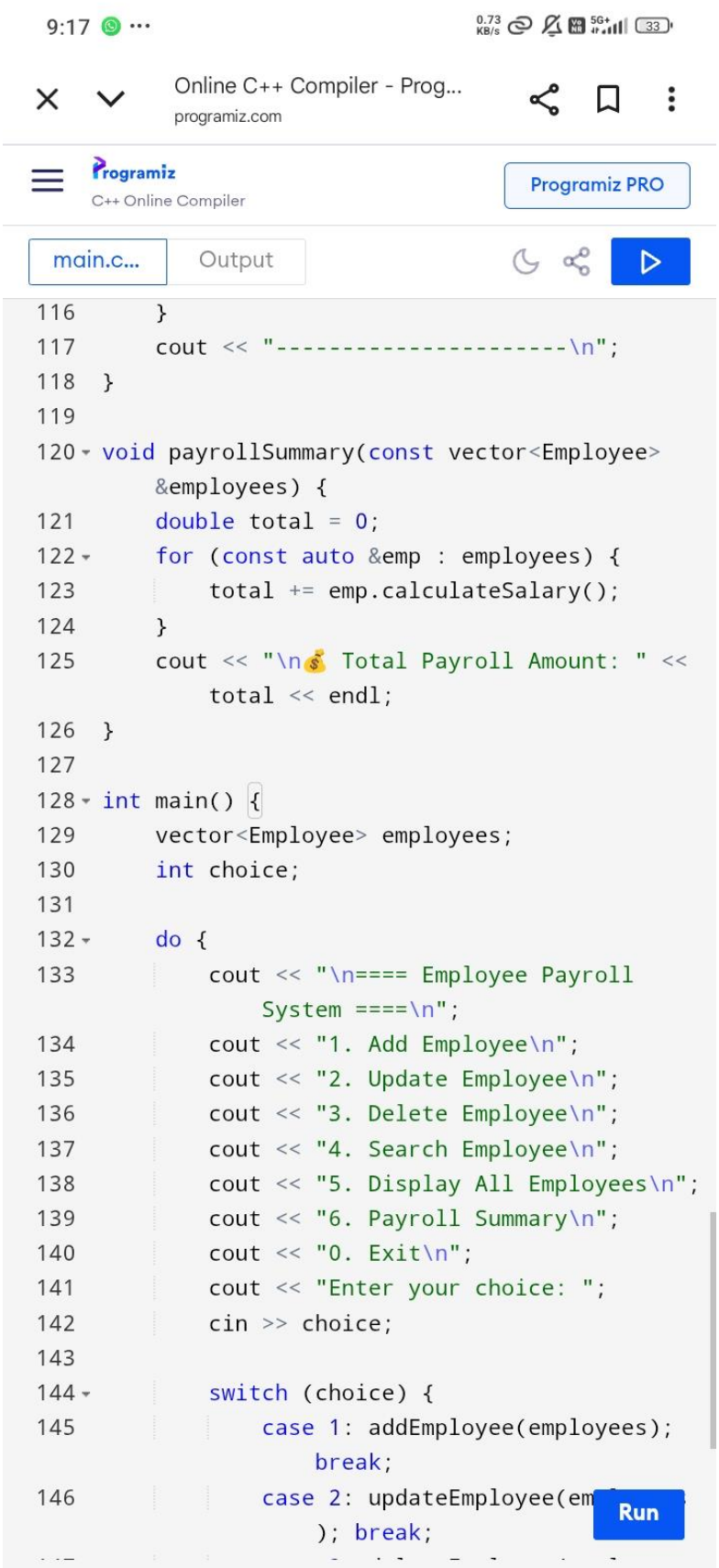

9:17
Online C++ Compiler - Prog...
programiz.com
Programiz
C++ Online Compiler
Programiz PRO
main.c...
Output
30 cout << "\n--- Salary Slip ---\n";
31 cout << "Employee ID : " << id << endl;
32 cout << "Employee Name : " << name << endl;
33 cout << "Hours Worked : " << hoursWorked << endl;
34 cout << "Hourly Rate : " << hourlyRate << endl;
35 cout << "Total Salary : " << calculateSalary() << endl;
36 cout << "--------------------\n";
37 }
38 };
39
40 // Utility Functions
41 void addEmployee(vector<Employee> &employees) {
42 string name;
43 int id;
44 double hours, rate;
45
46 cout << "Enter Employee ID: ";
47 cin >> id;
48 cin.ignore();
49 cout << "Enter Employee Name: ";
50 getline(cin, name);
51 cout << "Enter Hours Worked: ";
52 cin >> hours;
53 cout << "Enter Hourly Rate: ";
54 cin >> rate;
55
56 employees.emplace_back(name, id, hours, rate);
57 cout << "✅ Employee added successfully!\n";
Run




```cpp
    cout << "✅ Employee added
        successfully!\n";
}

void updateEmployee(vector<Employee>
    &employees) {
    int id;
    cout << "Enter Employee ID to update: ";
    cin >> id;

    for (auto &emp : employees) {
        if (emp.getId() == id) {
            double hours, rate;
            cout << "Enter New Hours Worked:
                ";
            cin >> hours;
            cout << "Enter New Hourly Rate: "
                ;
            cin >> rate;
            emp.setHoursWorked(hours);
            emp.setHourlyRate(rate);
            cout << "✅ Employee updated
                successfully!\n";
            return;
        }
    }
    cout << "❌ Employee not found!\n";
}

void deleteEmployee(vector<Employee>
    &employees) {
    int id;
    cout << "Enter Employee ID to delete: ";
    cin >> id;

    for (auto it = employees.begin();
```




```
    for (auto it = employees.begin(); it !=
        employees.end(); ++it) {
        if (it->getId() == id) {
            employees.erase(it);
            cout << "✅ Employee deleted
                successfully!\n";
            return;
        }
    }
    cout << "❌ Employee not found!\n";
}

void searchEmployee(const vector<Employee>
    &employees) {
    int id;
    cout << "Enter Employee ID to search: ";
    cin >> id;

    for (const auto &emp : employees) {
        if (emp.getId() == id) {
            emp.generateSlip();
            return;
        }
    }
    cout << "❌ Employee not found!\n";
}

void displayAllEmployees(const vector
    <Employee> &employees) {
    cout << "\n--- Employee List ---\n";
    for (const auto &emp : employees) {
        cout << "ID: " << emp.getId()
             << " | Name: " << emp.getName()
             << " | Salary: " << emp
                .calculateSalary() <
    }
```


9:17
Online C++ Compiler - Prog...
programiz.com
Programiz
C++ Online Compiler
Programiz PRO
main.c...
Output
116 }
117 cout << "----------------------\n";
118 }
119
120 void payrollSummary(const vector<Employee>
&employees) {
121 double total = 0;
122 for (const auto &emp : employees) {
123 total += emp.calculateSalary();
124 }
125 cout << "\n💰 Total Payroll Amount: " <<
total << endl;
126 }
127
128 int main() {
129 vector<Employee> employees;
130 int choice;
131
132 do {
133 cout << "\n==== Employee Payroll
System ====\n";
134 cout << "1. Add Employee\n";
135 cout << "2. Update Employee\n";
136 cout << "3. Delete Employee\n";
137 cout << "4. Search Employee\n";
138 cout << "5. Display All Employees\n";
139 cout << "6. Payroll Summary\n";
140 cout << "0. Exit\n";
141 cout << "Enter your choice: ";
142 cin >> choice;
143
144 switch (choice) {
145 case 1: addEmployee(employees);
break;
146 case 2: updateEmployee(em
); break;
Run


9:17
Online C++ Compiler - Prog...
programiz.com
Programiz
C++ Online Compiler
Programiz PRO
main.c...
Output
131
132 do {
133 cout << "\n==== Employee Payroll System ====\n";
134 cout << "1. Add Employee\n";
135 cout << "2. Update Employee\n";
136 cout << "3. Delete Employee\n";
137 cout << "4. Search Employee\n";
138 cout << "5. Display All Employees\n";
139 cout << "6. Payroll Summary\n";
140 cout << "0. Exit\n";
141 cout << "Enter your choice: ";
142 cin >> choice;
143
144 switch (choice) {
145 case 1: addEmployee(employees); break;
146 case 2: updateEmployee(employees); break;
147 case 3: deleteEmployee(employees); break;
148 case 4: searchEmployee(employees); break;
149 case 5: displayAllEmployees(employees); break;
150 case 6: payrollSummary(employees); break;
151 case 0: cout << "👋 Exiting...\n"; break;
152 default: cout << "❌ Invalid choice!\n"; break;
153 }
154 } while (choice != 0);
155
156 return 0;
157 }
Run


```
==== Employee Payroll System ====
1. Add Employee
2. Update Employee
3. Delete Employee
4. Search Employee
5. Display All Employees
6. Payroll Summary
0. Exit
Enter your choice: 0
👋 Exiting...


=== Code Execution Successful ===
```