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.
Program:

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

class Book {
private:
    int bookID;
    string title;
    string author;
    bool available;

public:
    Book(int id = 0, string t = "", string a = "", bool av = true) {
        bookID = id;
        title = t;
        author = a;
        available = av;
    }

    int getID() { return bookID; }
    string getTitle() { return title; }
    string getAuthor() { return author; }
    bool isAvailable() { 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 displayBook() {
    cout << "ID: " << bookID << " | Title: " << title
            << " | Author: " << author
            << " | Status: " << (available ? "Available" : "Issued") << endl;
    }
    string toFileString() {
    return to_string(bookID) + "," + title + "," + author + "," + (available ? "1" : "0") + "\n";
    }
    static Book fromFileString(string line) {
    int id;
    string t, a;
    bool av;
    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);
    av = (line.substr(pos3 + 1) == "1");

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

class Library {
private:
    vector<Book> books;
    string filename = "library.txt";
 void saveToFile() {
```

```
        ofstream fout(filename);
 for (auto &book : books) {
       fout << book.toFileString();
        }
        fout.close();
    }

 void loadFromFile() {
    books.clear();
    ifstream fin(filename);
    string line;
    while (getline(fin, line)) {
    if (!line.empty()) {
        books.push_back(Book::fromFileString(line));
            }
        }
        fin.close();
    }

public:
    Library() {
        loadFromFile();
    }

    void addBook(int id, string title, string author) {
    books.push_back(Book(id, title, author));
        saveToFile();
     cout << "Book added successfully!\n";
    }

 void issueBook(int id) {
 for (auto &book : books) {
 if (book.getID() == id) {
      book.issueBook();
      saveToFile();
      return;
        }
     }
     cout << "Book not found!\n";
    }

 void returnBook(int id) {
    for (auto &book : books) {
    if (book.getID() == id) {
```

```
        book.returnBook();
        saveToFile();
        return;
           }
        }
     cout << "Book not found!\n";
   }

void searchByID(int id) {
   for (auto &book : books) {
   if (book.getID() == id) {
      book.displayBook();
        return;
        }
      }
      cout << "Book not found!\n";
   }

 void searchByTitle(string title) {
    for (auto &book : books) {
    if (book.getTitle() == title) {
        book.displayBook();
           return;
        }
     }
        cout << "Book not found!\n";
   }

 void displayAllBooks() {
        cout << "\n--- Library Collection ---\n";
  for (auto &book : books) {
           book.displayBook();
        }
   }

   void displayStats() {
        int total = books.size();
        int issued = 0, available = 0;
    for (auto &book : books) {
    if (book.isAvailable())
              available++;
    else
              issued++;
        }
```

```
    cout << "\n--- Library Stats ---\n";
    cout << "Total Books: " << total << endl;
    cout << "Available: " << available << endl;
    cout << "Issued: " << issued << endl;
   }
};

int main() {
   Library library;
   int choice;

 do {
    cout << "\n====== Library Management Menu ======\n";
    cout << "1. Add Book\n";
    cout << "2. Issue Book\n";
    cout << "3. Return Book\n";
    cout << "4. Search Book by ID\n";
    cout << "5. Search Book by Title\n";
    cout << "6. Display All Books\n";
    cout << "7. Display Stats\n";
    cout << "8. Exit\n";
    cout << "Enter your choice: ";
    cin >> choice;

    int id;
    string title, author;

    switch (choice) {
    case 1:
       cout << "Enter Book ID: ";
       cin >> id;
       cout << "Enter Title: ";
       cin.ignore();
       getline(cin, title);
       cout << "Enter Author: ";
       getline(cin, author);
       library.addBook(id, title, author);
       break;

     case 2:
       cout << "Enter Book ID to Issue: ";
       cin >> id;
       library.issueBook(id);
       break;
```

```
      case 3:
        cout << "Enter Book ID to Return: ";
        cin >> id;
        library.returnBook(id);
        break;

      case 4:
        cout << "Enter Book ID: ";
        cin >> id;
        library.searchByID(id);
        break;

      case 5:
         cout << "Enter Title: ";
         cin.ignore();
         getline(cin, title);
         library.searchByTitle(title);
          break;

      case 6:
         library.displayAllBooks();
         break;

      case 7:
         library.displayStats();
         break;

      case 8:
         cout << "Exiting Library System...\n";
         break;

      default:
         cout << "Invalid choice! Try again.\n";
       }
    } while (choice != 8);

    return 0;
}
```

O/p

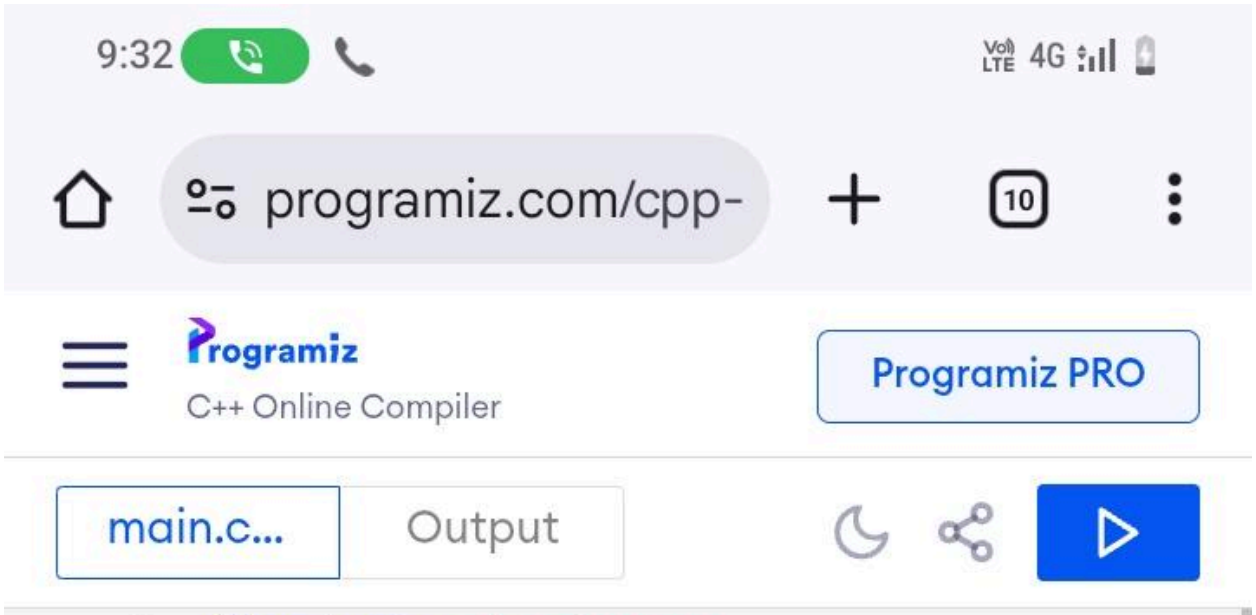
```
====== Library Management Menu ======
1. Add Book
2. Issue Book
3. Return Book
```

4. Search Book by ID
5. Search Book by Title
6. Display All Books
7. Display Stats
8. Exit
Enter your choice: 6

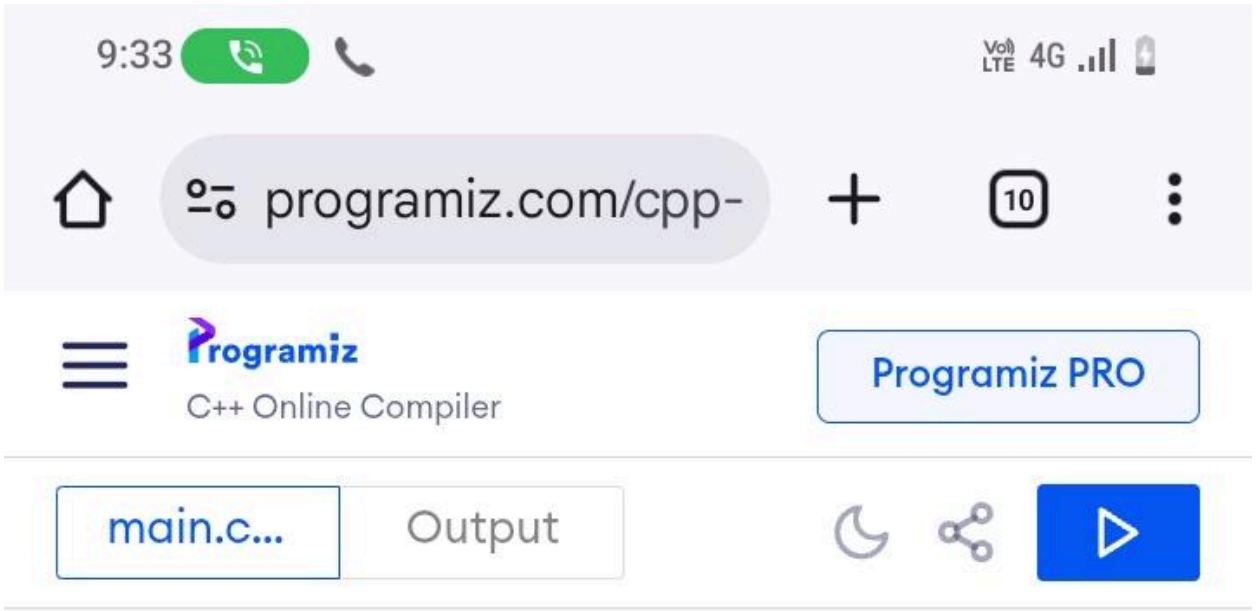
--- Library Collection ---

====== Library Management Menu ======
1. Add Book
2. Issue Book
3. Return Book
4. Search Book by ID
5. Search Book by Title
6. Display All Books
7. Display Stats
8. Exit
Enter your choice: 8

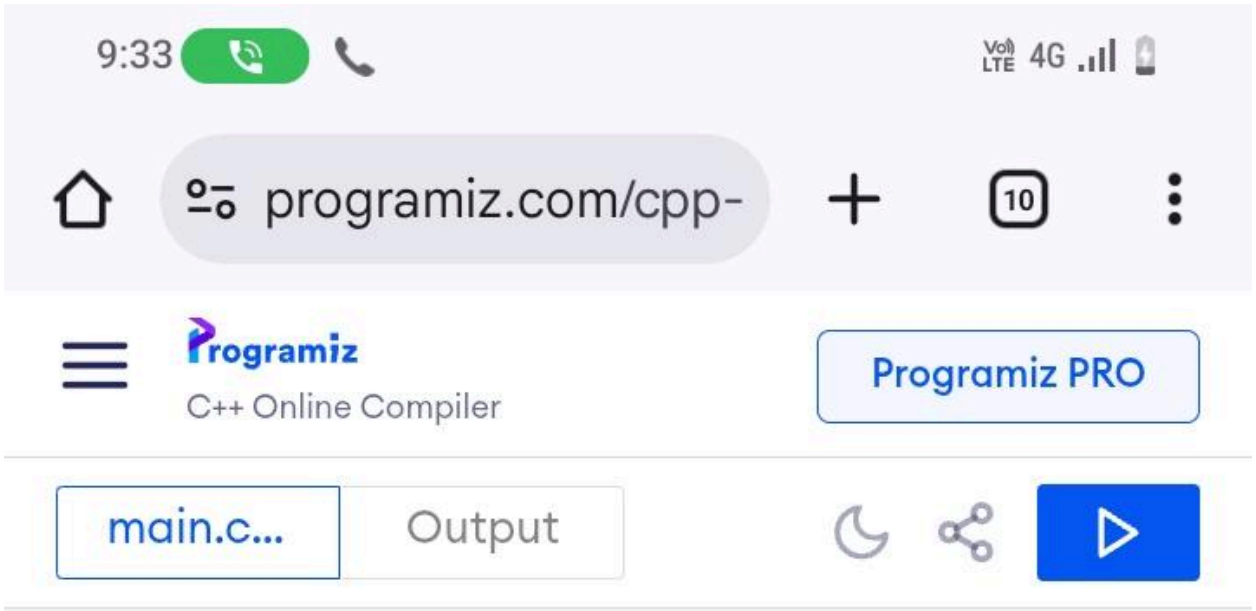
Exiting Library System…



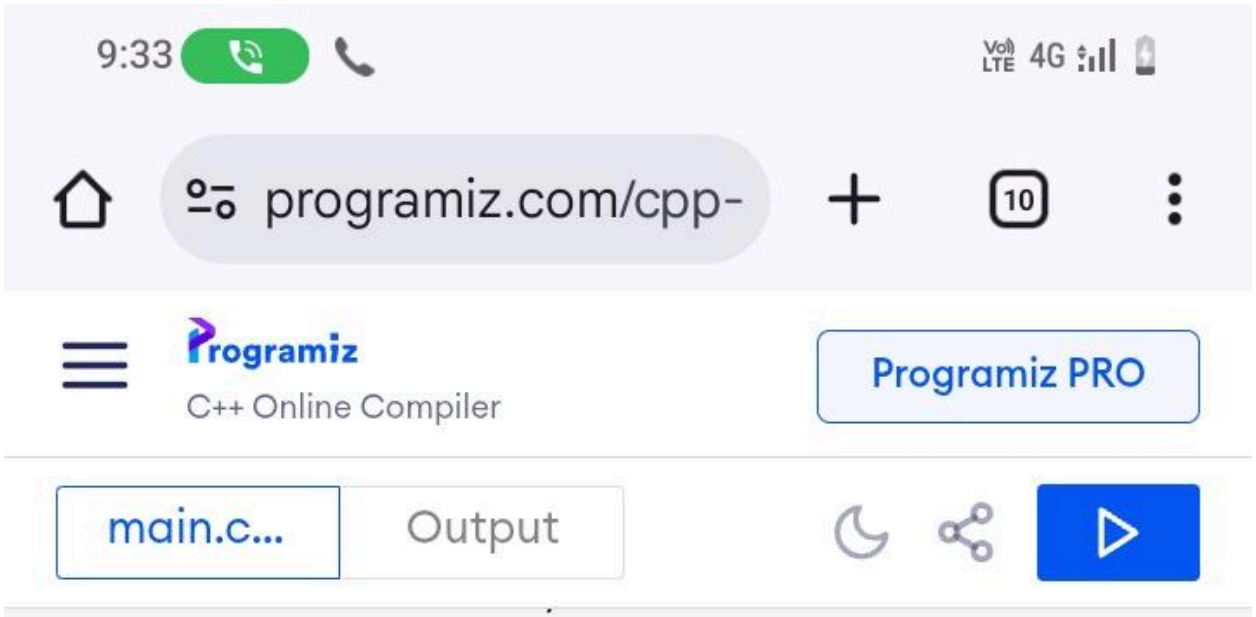

```cpp
#include <iostream>
#include <vector>
#include <fstream>
#include <string>
using namespace std;

class Book {
private:
    int bookID;
    string title;
    string author;
    bool available;

public:
    Book(int id = 0, string t = "",
         string a = "", bool av = true
         ) {
        bookID = id;
```



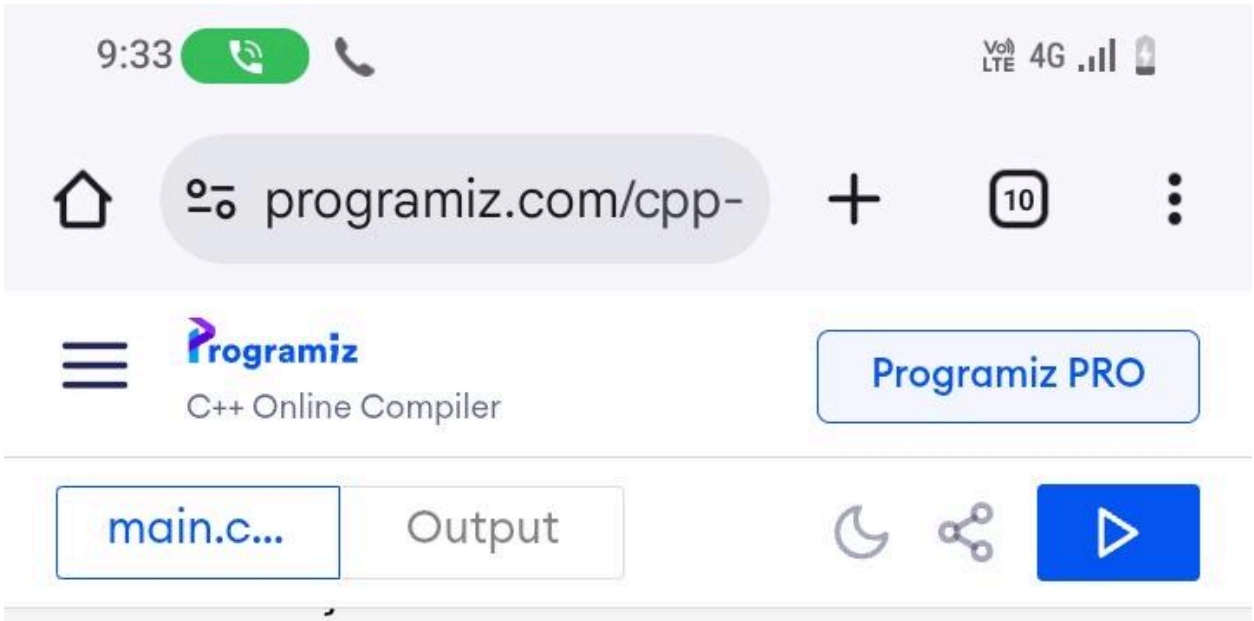

```
25      bool isAvailable() { return
            available; }
26
27      void issueBook() {
28      if (available) {
29        available = false;
30          cout << "Book issued
                successfully!\n";
31          }
32      else {
33          cout << "Book is already
                issued.\n";
34          }
35      }
36
37      void returnBook() {
38      if (!available) {
39      available = true;
```



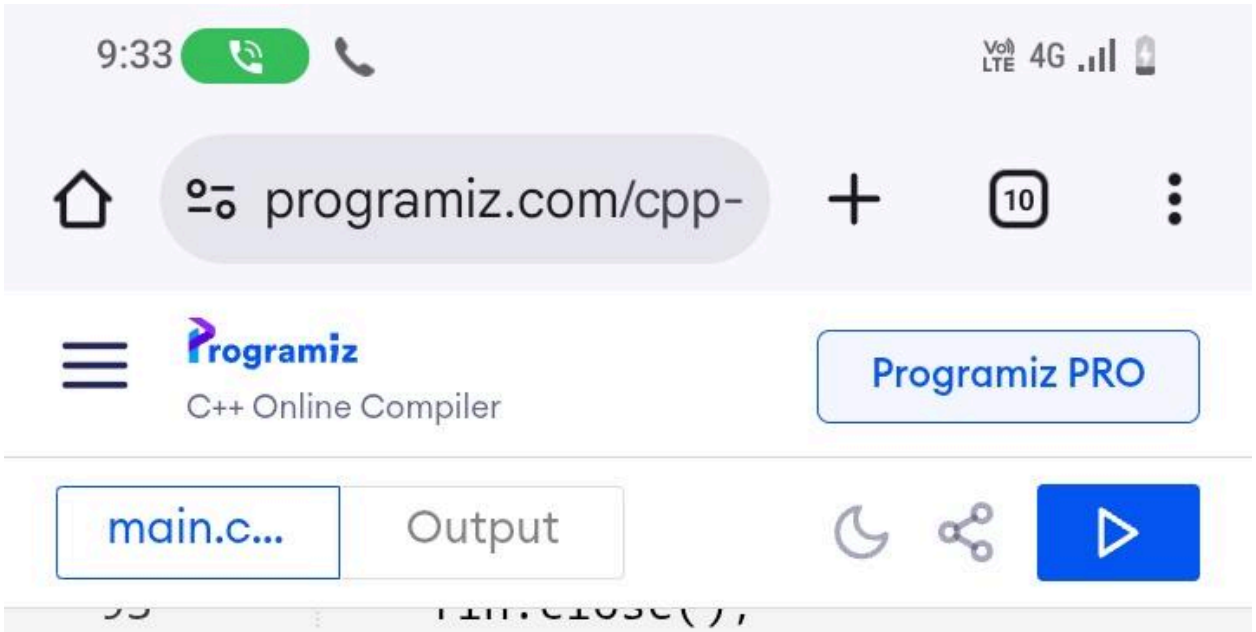

```
43          cout << "Book was not issued
                .\n";
44          }
45      }
46
47      void displayBook() {
48      cout << "ID: " << bookID << " |
            Title: " << title
49                  << " | Author: " <<
                        author
50                  << " | Status: " <<
                        (available ?
                        "Available" :
                        "Issued") << endl;
51      }
52      string toFileString() {
53      return to_string(bookID) + "," +
            title + "," + author + "," +
```




```
54      }
55      static Book fromFileString(string
            line) {
56      int id;
57      string t, a;
58      bool av;
59      size_t pos1 = line.find(",");
60      size_t pos2 = line.find(",", pos1
            + 1);
61      size_t pos3 = line.find(",", pos2
            + 1);
62
63      id = stoi(line.substr(0, pos1));
64      t = line.substr(pos1 + 1, pos2 -
            pos1 - 1);
65      a = line.substr(pos2 + 1, pos3 -
            pos2 - 1);
66      av = (line.substr(pos3 + 1) ==
```

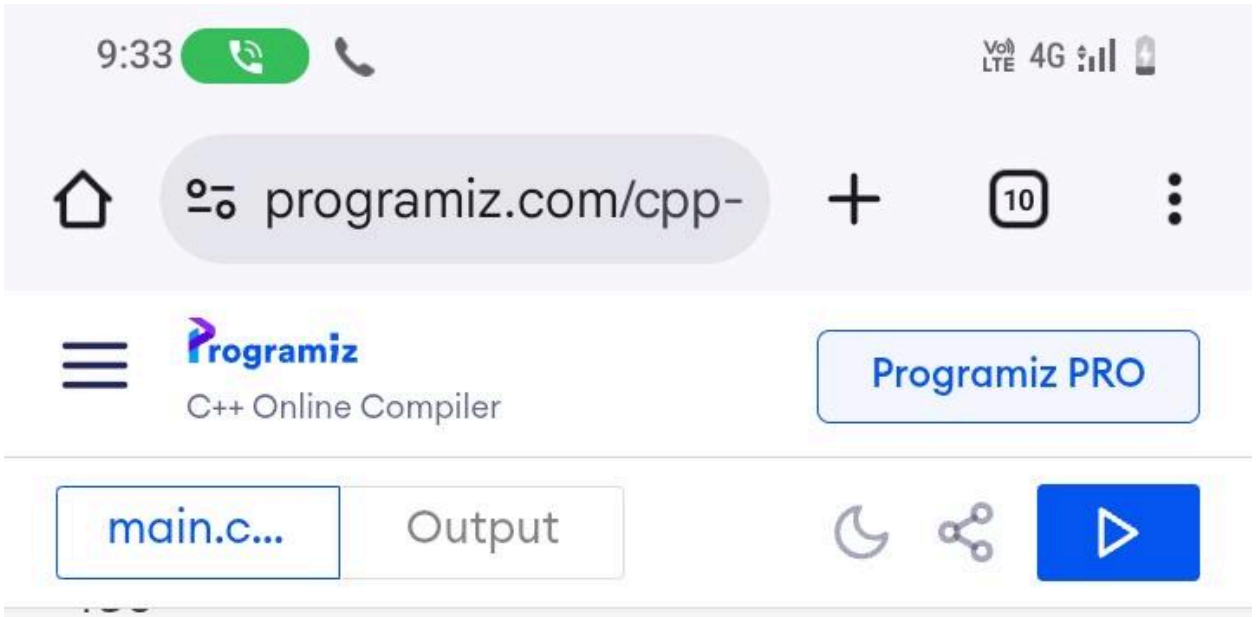



```
70  };
71
72  class Library {
73  private:
74      vector<Book> books;
75      string filename = "library.txt";
76   void saveToFile() {
77          ofstream fout(filename);
78    for (auto &book : books) {
79         fout << book.toFileString();
80          }
81          fout.close();
82      }
83
84   void loadFromFile() {
85      books.clear();
86      ifstream fin(filename);
87      string line;
```

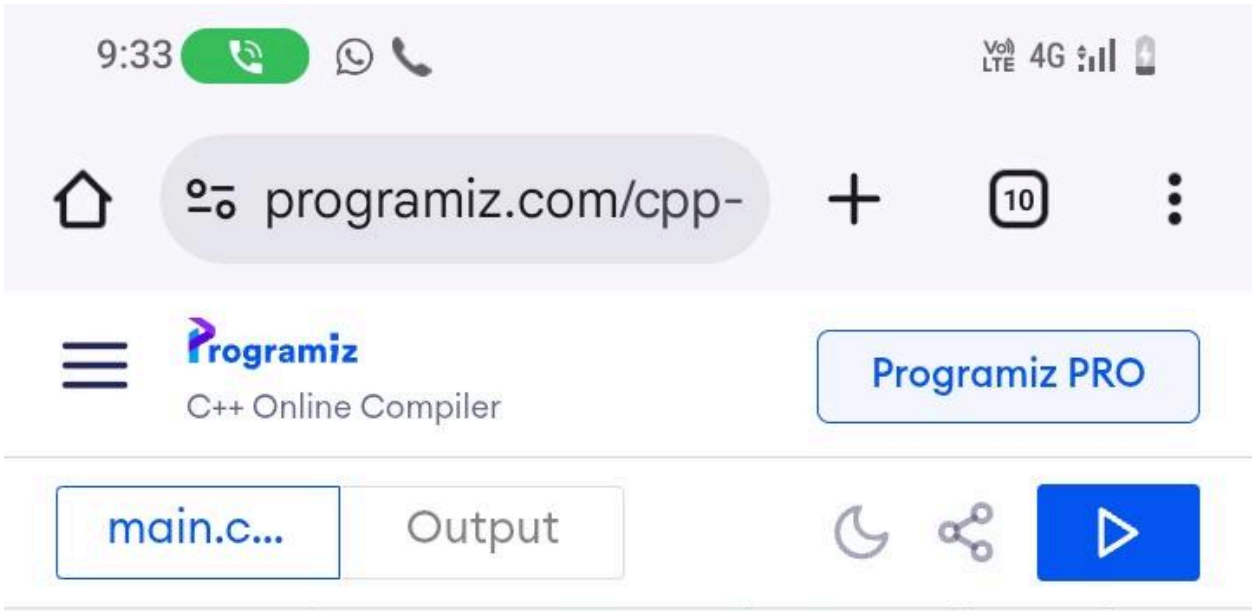



```
94      }
95
96  public:
97      Library() {
98          loadFromFile();
99      }
100
101     void addBook(int id, string title
            , string author) {
102     books.push_back(Book(id, title,
            author));
103         saveToFile();
104      cout << "Book added
             successfully!\n";
105     }
106
107 void issueBook(int id) {
108 for (auto &book : books) {
```

```
114         }
115         cout << "Book not found!\n";
116     }
117
118 void returnBook(int id) {
119     for (auto &book : books) {
120     if (book.getID() == id) {
121         book.returnBook();
122         saveToFile();
123         return;
124             }
125         }
126         cout << "Book not found!\n";
127     }
128
129 void searchByID(int id) {
130     for (auto &book : books) {
131     if (book.getID() == id) {
132         book.displayBook();
```

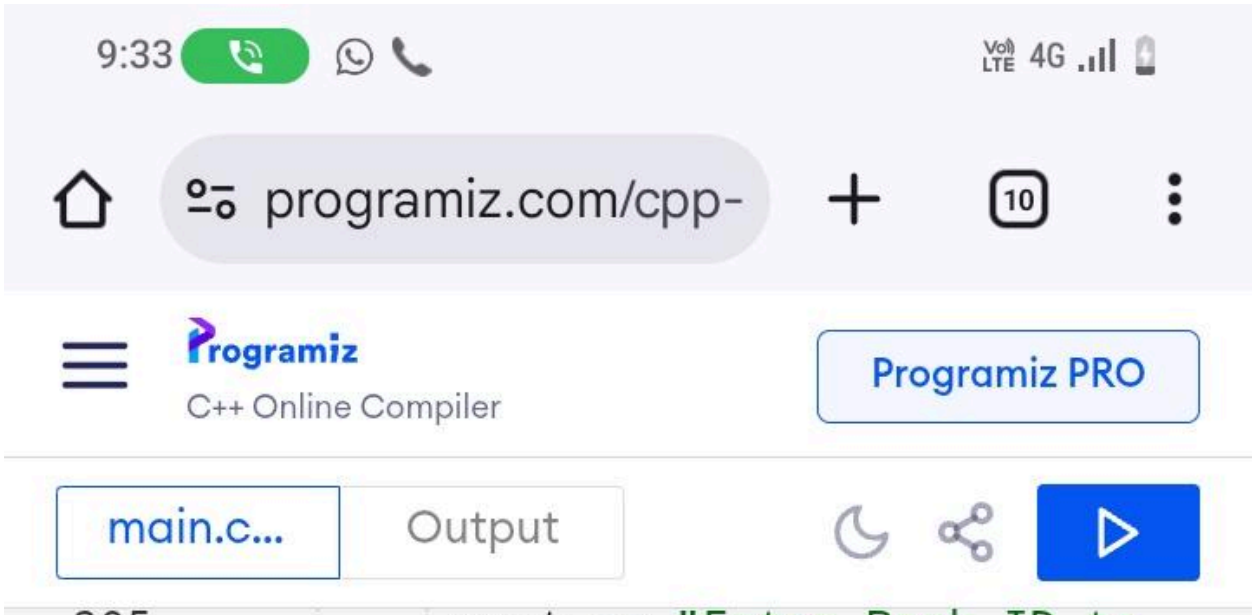



```
139   void searchByTitle(string title) {
140       for (auto &book : books) {
141       if (book.getTitle() == title) {
142           book.displayBook();
143               return;
144           }
145       }
146           cout << "Book not found!\n";
147       }
148
149   void displayAllBooks() {
150           cout << "\n--- Library
                  Collection ---\n";
151       for (auto &book : books) {
152               book.displayBook();
153           }
154       }
155
```

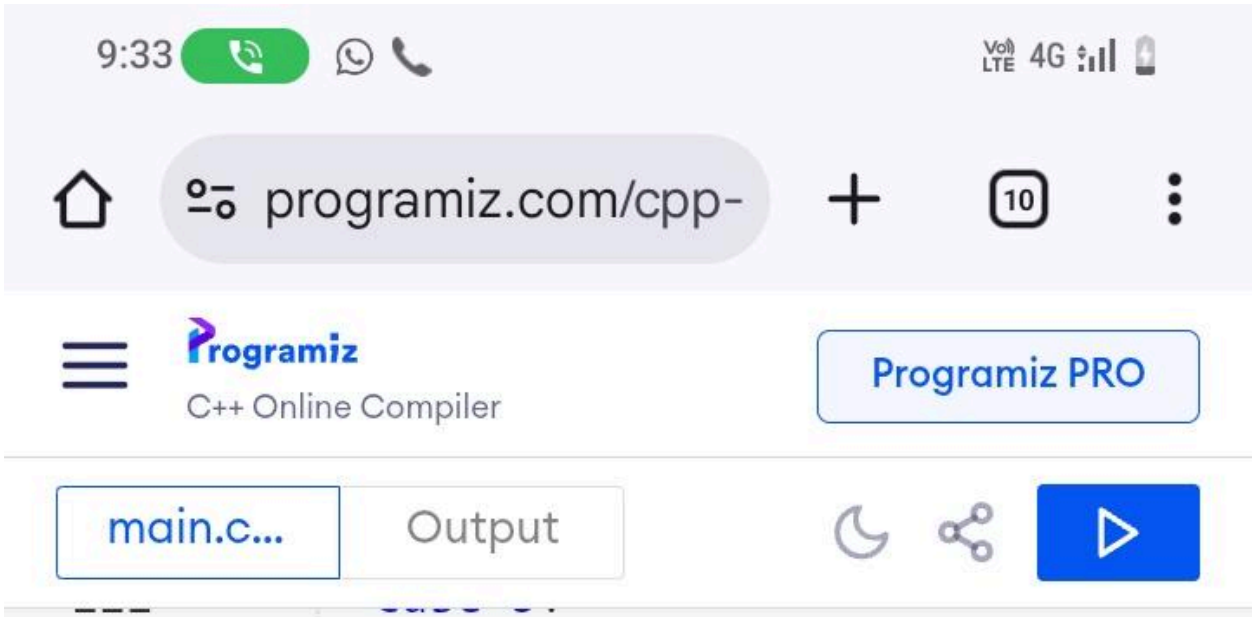



```
168         cout << "Issued: " << issued <<
                endl;
169     }
170 };
171
172 int main() {
173     Library library;
174     int choice;
175
176     do {
177         cout << "\n====== Library
                Management Menu ======\n";
178         cout << "1. Add Book\n";
179         cout << "2. Issue Book\n";
180         cout << "3. Return Book\n";
181         cout << "4. Search Book by ID\n"
                ;
182         cout << "5. Search Book by
                Title\n";
```

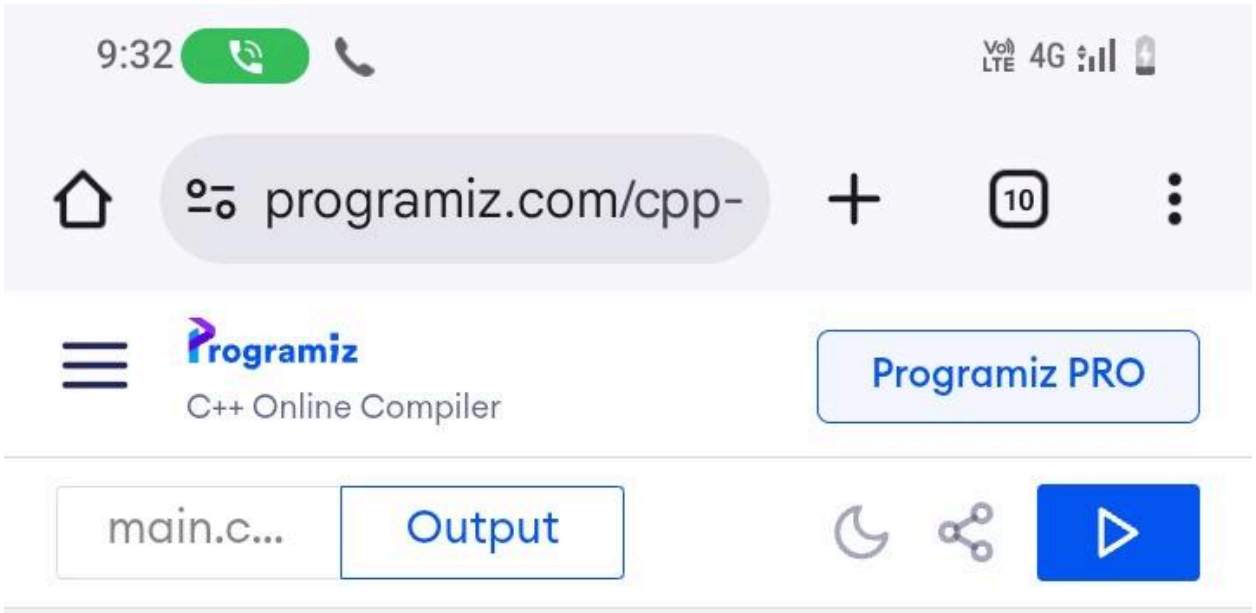
```
187     cin >> choice;
188
189     int id;
190     string title, author;
191
192     switch (choice) {
193     case 1:
194         cout << "Enter Book ID: ";
195         cin >> id;
196         cout << "Enter Title: ";
197         cin.ignore();
198         getline(cin, title);
199         cout << "Enter Author: ";
200         getline(cin, author);
201         library.addBook(id, title,
                author);
202         break;
203
204      case 2:
```




```
205             cout << "Enter Book ID to
                    Issue: ";
206             cin >> id;
207             library.issueBook(id);
208             break;
209
210         case 3:
211             cout << "Enter Book ID to
                    Return: ";
212             cin >> id;
213             library.returnBook(id);
214             break;
215
216         case 4:
217             cout << "Enter Book ID: ";
218             cin >> id;
219             library.searchByID(id);
220             break;
221
```




```
223             cout << "Enter Title: ";
224             cin.ignore();
225             getline(cin, title);
226             library.searchByTitle(title
                    );
227              break;
228
229         case 6:
230             library.displayAllBooks();
231             break;
232
233         case 7:
234             library.displayStats();
235             break;
236
237         case 8:
238             cout << "Exiting Library
                    System...\n";
```




```
====== Library Management Menu ======
1. Add Book
2. Issue Book
3. Return Book
4. Search Book by ID
5. Search Book by Title
6. Display All Books
7. Display Stats
8. Exit
Enter your choice: 6

--- Library Collection ---

====== Library Management Menu ======
1. Add Book
2. Issue Book
3. Return Book
```

My view:
stores, tracks, and updates books automatically

provides information quickly and reliably

represents real-world library operations in an OOP way

Objective:

The code is built to manage a library's collection of books digitally.
It simplifies adding, issuing, returning, and searching for books.
Add new books to the system.
Issue and return books with status updates.
Search books by ID or title.
Show statistics: total books, issued books, and available books.
To replace manual record-keeping with a faster, reliable, and automated solution.
Provides a digital librarian's role.
Reduces errors and saves time.
Easy for both librarians and students to access information.

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.

Program:

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

class Employee {
private:
    int id;
    string name;
    float hoursWorked;
```

```
    float hourlyRate;

public:
    Employee(int i, string n, float h, float r) {
        id = i;
        name = n;
        hoursWorked = h;
        hourlyRate = r;
    }

    int getID() { return id; }
    string getName() { return name; }
    float calculateSalary() { return hoursWorked * hourlyRate; }

    void displaySalarySlip() {
        cout << "\n--- Salary Slip ---\n";
        cout << "ID: " << id << "\nName: " << name
             << "\nHours Worked: " << hoursWorked
             << "\nHourly Rate: " << hourlyRate
             << "\nTotal Salary: " << calculateSalary() << "\n";
    }

    void update(float h, float r) {
        hoursWorked = h;
        hourlyRate = r;
    }
};

class PayrollSystem {
private:
    vector<Employee> employees;

public:
    void addEmployee(int id, string name, float hours, float rate) {
        employees.push_back(Employee(id, name, hours, rate));
        cout << "Employee added successfully!\n";
    }

    void updateEmployee(int id, float hours, float rate) {
        for (auto &e : employees) {
            if (e.getID() == id) {
                e.update(hours, rate);
                cout << "Employee updated successfully!\n";
                return;
```

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

    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";
    }

    void searchEmployee(int id) {
        for (auto &e : employees) {
            if (e.getID() == id) {
                e.displaySalarySlip();
                return;
            }
        }
        cout << "Employee not found!\n";
    }

    void listEmployees() {
        cout << "\n--- Employee List ---\n";
        for (auto &e : employees) {
            cout << "ID: " << e.getID() << " | Name: " << e.getName()
                 << " | Salary: " << e.calculateSalary() << endl;
        }
    }

    void summaryReport() {
        float total = 0;
        for (auto &e : employees) {
            total += e.calculateSalary();
        }
        cout << "\n--- Payroll Summary ---\n";
        cout << "Total Payroll Amount: " << total << endl;
    }
};
```

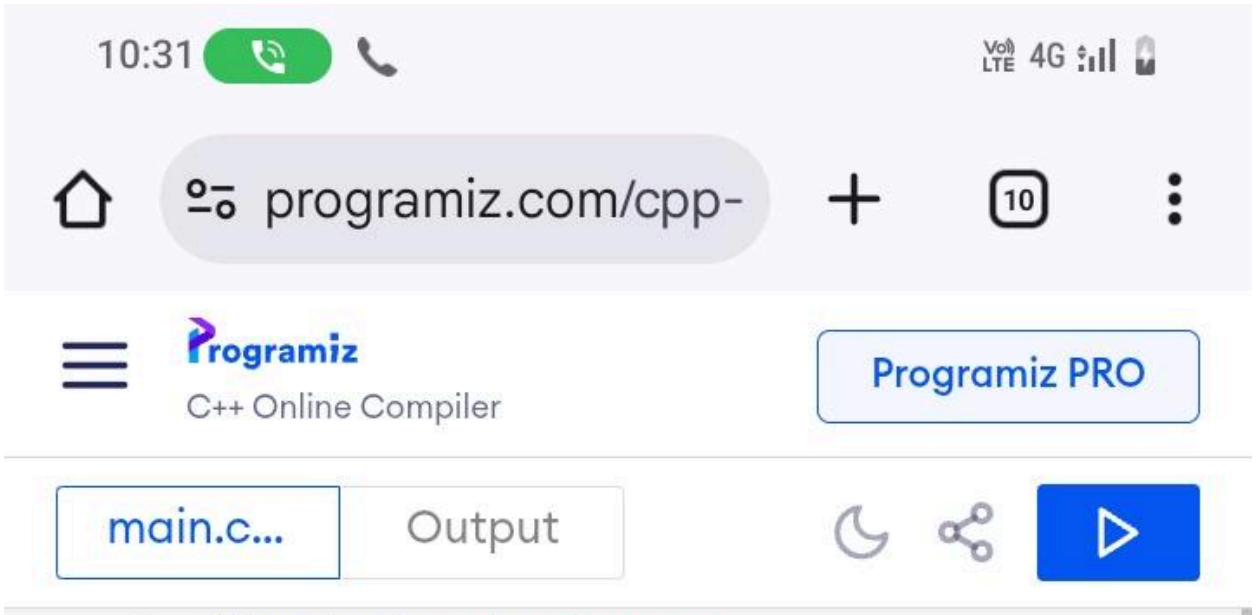
```
int main() {
    PayrollSystem system;
    int choice, id;
    string name;
    float hours, rate;

    do {
        cout << "\n===== Employee Payroll Menu =====\n";
        cout << "1. Add Employee\n2. Update Employee\n3. Delete Employee\n";
        cout << "4. Search Employee\n5. List Employees\n6. Summary Report\n7. Exit\n";
        cout << "Enter choice: ";
        cin >> choice;

        switch (choice) {
        case 1:
            cout << "Enter ID: "; cin >> id;
            cout << "Enter Name: "; cin.ignore(); getline(cin, name);
            cout << "Enter Hours Worked: "; cin >> hours;
            cout << "Enter Hourly Rate: "; cin >> rate;
            system.addEmployee(id, name, hours, rate);
            break;
        case 2:
            cout << "Enter ID to Update: "; cin >> id;
            cout << "Enter New Hours: "; cin >> hours;
            cout << "Enter New Rate: "; cin >> rate;
            system.updateEmployee(id, hours, rate);
            break;
        case 3:
            cout << "Enter ID to Delete: "; cin >> id;
            system.deleteEmployee(id);
            break;
        case 4:
            cout << "Enter ID to Search: "; cin >> id;
            system.searchEmployee(id);
            break;
        case 5:
            system.listEmployees();
            break;
        case 6:
            system.summaryReport();
            break;
        case 7:
            cout << "Exiting Payroll System...\n";
            break;
```

```
        default:
            cout << "Invalid choice! Try again.\n";
        }
    } while (choice != 7);

    return 0;
}
```

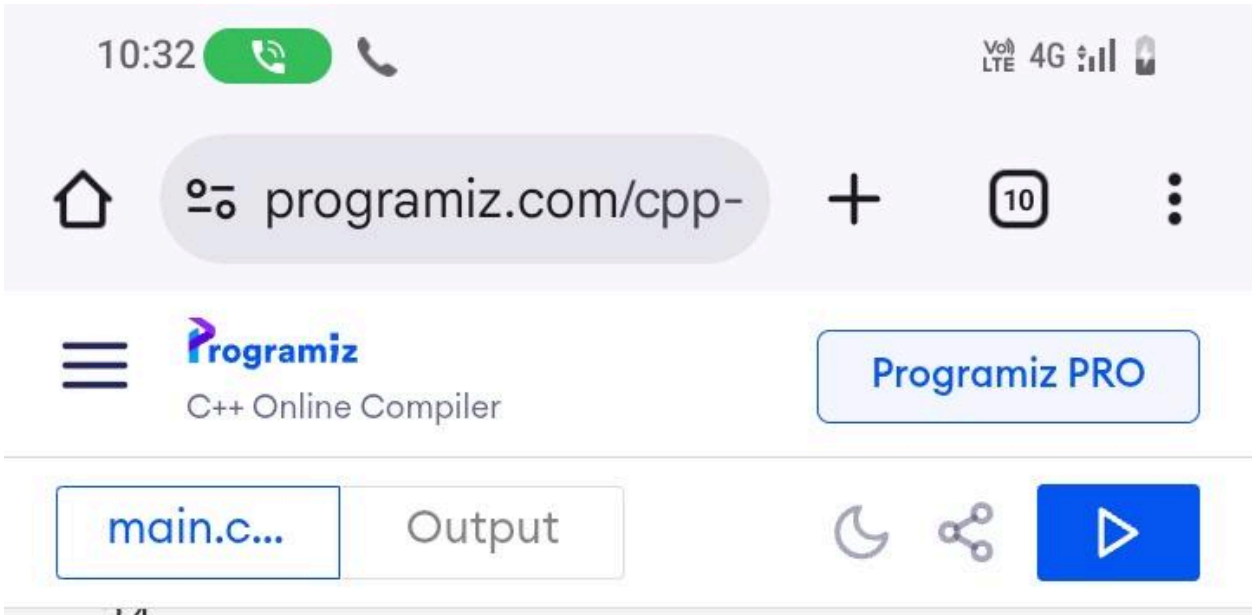
O/p

```
===== Employee Payroll Menu =====
1. Add Employee
2. Update Employee
3. Delete Employee
4. Search Employee
5. List Employees
6. Summary Report
7. Exit
Enter choice: 1
Enter ID: 101
Enter Name: kiruba
Enter Hours Worked: 3
Enter Hourly Rate: 100
Employee added successfully!

===== Employee Payroll Menu =====
1. Add Employee
2. Update Employee
3. Delete Employee
4. Search Employee
5. List Employees
6. Summary Report
7. Exit
Enter choice: 7
```

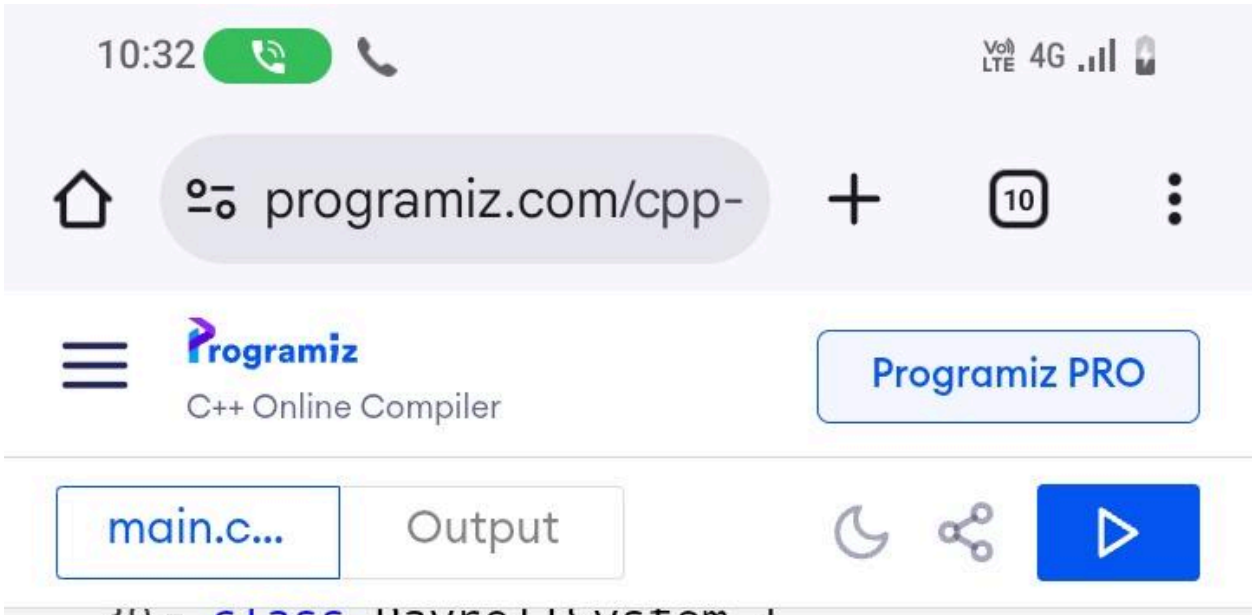
Exiting Payroll System…



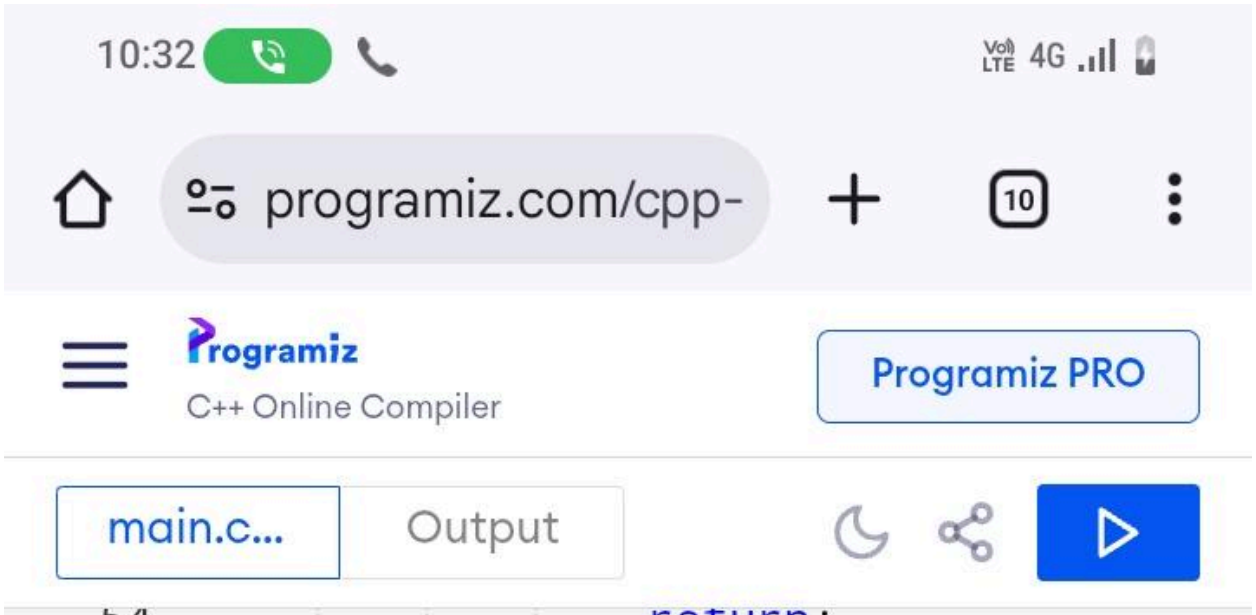

```cpp
#include <iostream>
#include <vector>
#include <string>
using namespace std;

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

public:
    Employee(int i, string n, float h
        , float r) {
        id = i;
        name = n;
        hoursWorked = h;
```



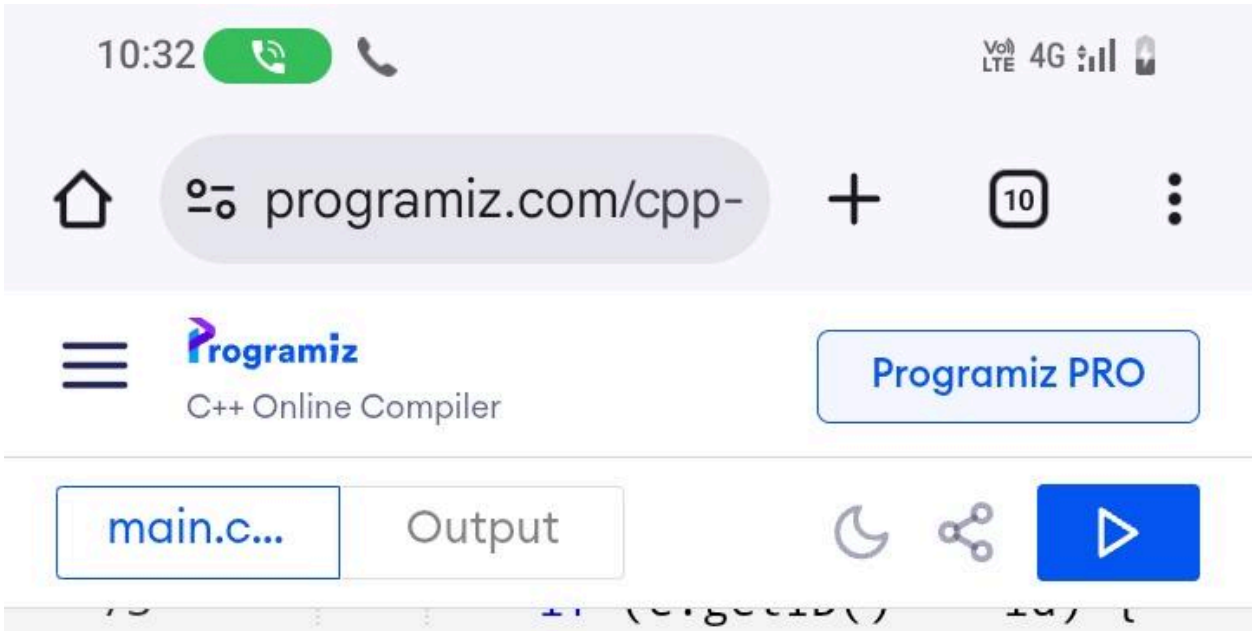

```
24
25      void displaySalarySlip() {
26          cout << "\n--- Salary Slip
                ---\n";
27          cout << "ID: " << id <<
                "\nName: " << name
28              << "\nHours Worked: " <<
                    hoursWorked
29              << "\nHourly Rate: " <<
                    hourlyRate
30              << "\nTotal Salary: " <<
                    calculateSalary() <<
                    "\n";
31      }
32
33      void update(float h, float r) {
34          hoursWorked = h;
35          hourlyRate = r;
36      }
```



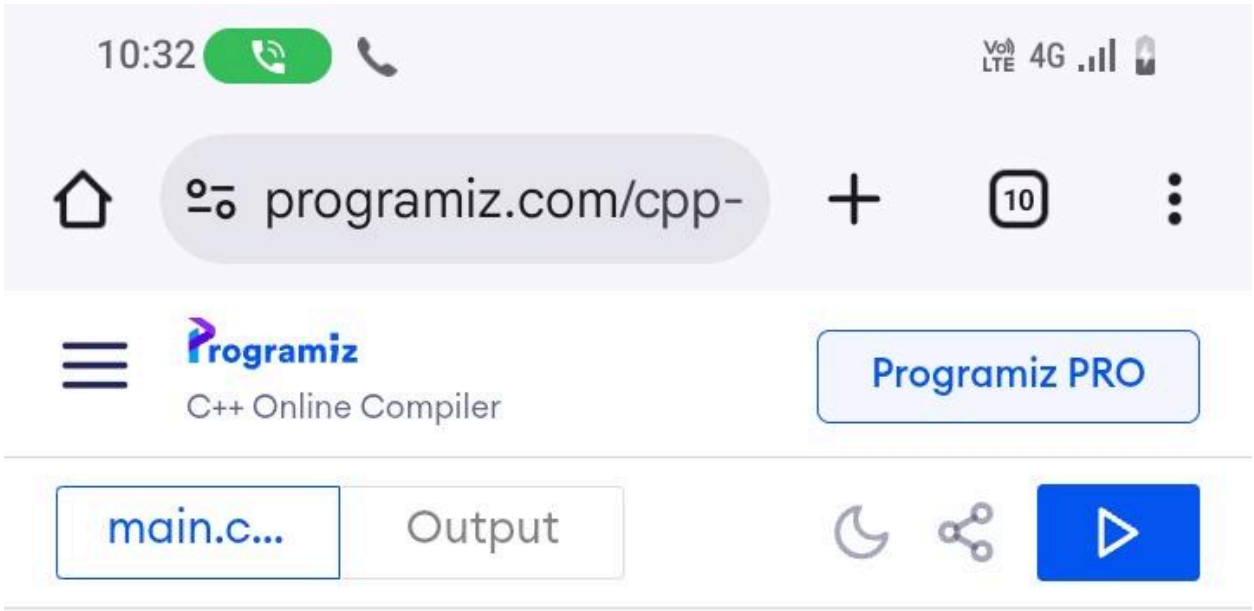

```
39 class PayrollSystem {
40 private:
41     vector<Employee> employees;
42
43 public:
44     void addEmployee(int id, string
          name, float hours, float rate
          ) {
45         employees.push_back(Employee
              (id, name, hours, rate));
46         cout << "Employee added
              successfully!\n";
47     }
48
49     void updateEmployee(int id, float
          hours, float rate) {
50         for (auto &e : employees) {
51             if (e.getID() == id) {
52                 e.update(hours, rate
```



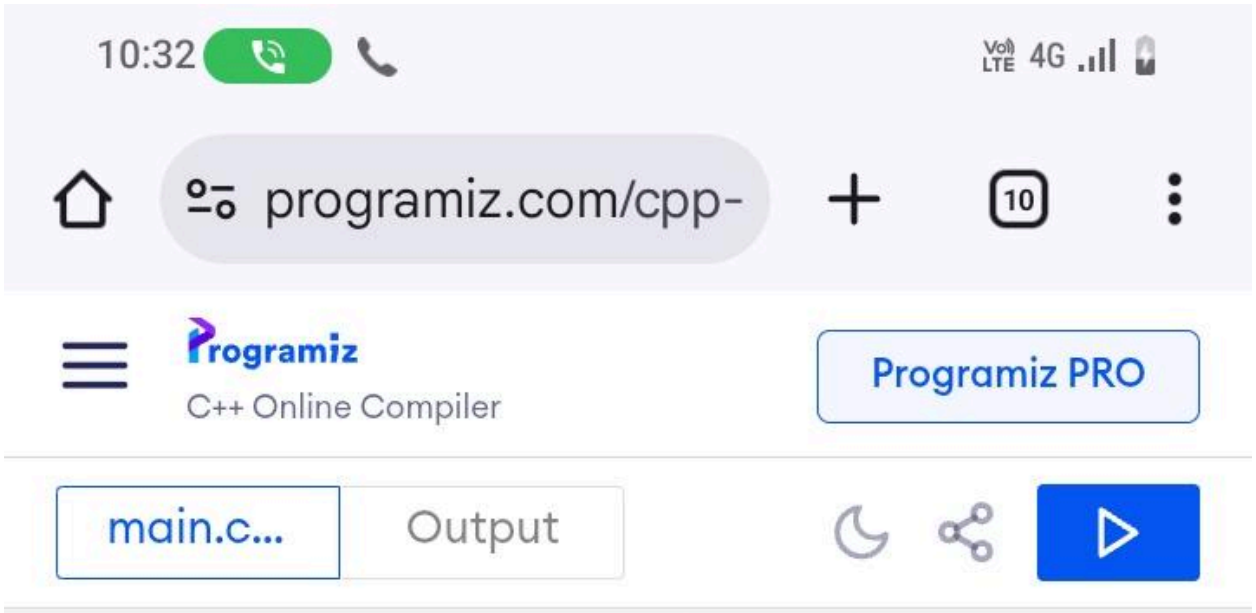

```
54                return;
55            }
56        }
57        cout << "Employee not
              found!\n";
58    }
59
60    void deleteEmployee(int id) {
61        for (auto it = employees
              .begin(); it != employees
              .end(); ++it) {
62            if (it->getID() == id) {
63                employees.erase(it);
64                cout << "Employee
                    deleted
                    successfully!\n";
65                return;
66            }
67        }
```




```
74                  e.displaySalarySlip
                        ();
75                  return;
76              }
77          }
78          cout << "Employee not
                found!\n";
79      }
80
81      void listEmployees() {
82          cout << "\n--- Employee List
                ---\n";
83          for (auto &e : employees) {
84              cout << "ID: " << e.getID
                    () << " | Name: " <<
                    e.getName()
85                   << " | Salary: " <<
                      e.calculateSalary()
```



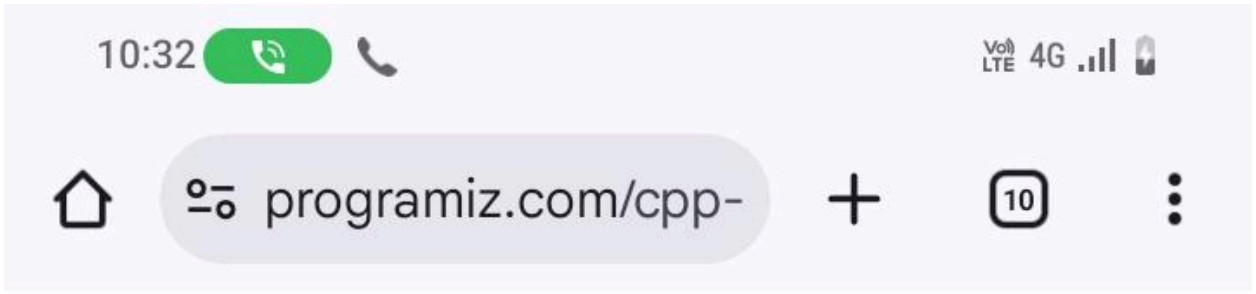

```
88
89      void summaryReport() {
90          float total = 0;
91          for (auto &e : employees) {
92              total += e
                    .calculateSalary();
93          }
94          cout << "\n--- Payroll
                Summary ---\n";
95          cout << "Total Payroll Amount
                : " << total << endl;
96      }
97  };
98
99  int main() {
100     PayrollSystem system;
101     int choice, id;
102     string name;
```




```
            Payroll Menu =====\n";
107         cout << "1. Add Employee\n2.
                Update Employee\n3.
                Delete Employee\n";
108         cout << "4. Search
                Employee\n5. List
                Employees\n6. Summary
                Report\n7. Exit\n";
109         cout << "Enter choice: ";
110         cin >> choice;
111
112         switch (choice) {
113         case 1:
114             cout << "Enter ID: "; cin
                    >> id;
115             cout << "Enter Name: ";
                    cin.ignore(); getline
                    (cin, name);
```

```
124             system.updateEmployee(id,
                    hours, rate);
125             break;
126         case 3:
127             cout << "Enter ID to
                    Delete: "; cin >> id;
128             system.deleteEmployee(id
                    );
129             break;
130         case 4:
131             cout << "Enter ID to
                    Search: "; cin >> id;
132             system.searchEmployee(id
                    );
133             break;
134         case 5:
135             system.listEmployees();
136             break;
137         case 6:
```

```
                Delete: "; cin >> id;
128             system.deleteEmployee(id
                );
129             break;
130         case 4:
131             cout << "Enter ID to
                Search: "; cin >> id;
132             system.searchEmployee(id
                );
133             break;
134         case 5:
135             system.listEmployees();
136             break;
137         case 6:
138             system.summaryReport();
139             break;
140         case 7:
141             cout << "Exiting Payroll
```



Programiz
C++ Online Compiler

Programiz PRO

main.c... Output

```
===== Employee Payroll Menu =====
1. Add Employee
2. Update Employee
3. Delete Employee
4. Search Employee
5. List Employees
6. Summary Report
7. Exit
Enter choice: 1
Enter ID: 101
Enter Name: kiruba
Enter Hours Worked: 3
Enter Hourly Rate: 100
Employee added successfully!

===== Employee Payroll Menu =====
1. Add Employee
2. Update Employee
```

Objective:
The code works like a digital payroll officer, ensuring accurate salaries, easy record management, and reduced workload.

My view:

To calculate employee salaries based on working hours and hourly rate.
To store and manage employee details like name, ID, hours worked, and pay rate.
To generate salary slips for employees. To allow adding, updating, searching, and deleting employee records.To digitize payroll management, reducing manual work and errors.