Object-oriented programming (OOPs)in C++

PROJECT NAME*:* Student Record Management System.

Submitted by: Arushi Saxena(24BCA10078)

Harjas Ahuja(24BCA10084)

Class: 24BCA-1(B)

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to everyone who contributed to the successful completion of this project.

First and foremost, I thank my instructor, Mrs. Konika mam, for their invaluable guidance and support throughout this project. Their insights and feedback were instrumental in enhancing my understanding of OOPs concept using C++.

I would also like to extend my appreciation to my classmates and friends who provided encouragement and assistance during the development process. Collaborating with them made the experience more enriching.

Finally, I am grateful to my family for their unwavering support and understanding during the course of this project.

Thank you all for your contributions!

By Arushi Saxena(24BCA10078)

Harjas Ahuja(24BCA10084)

CERTIFICATE

This is to certify that

Arushi Saxena and Harjas Ahuja

has successfully completed the OOPs using C++ Programming Project titled

Student Record Management System

during the first semester at the University Institute of Computing (UIC).

The project involved demonstrating proficiency in C programming and problem-solving skills.

Date:

Signature: _____

Mrs. Konika mam

INDEX

INTRODUCTION OF THE PROJECT

A **Student Record Management System (SRMS)** is a vital administrative tool that forms the digital backbone of any educational institution, moving beyond cumbersome paper-based methods to offer **efficient, centralized data handling**. This particular SRMS project, crafted using **C++ and Object-Oriented Programming (OOP)**, is designed to perform the full spectrum of data management tasks: creating, viewing, searching, updating, and deleting student records. It features a straightforward **menu-driven interface** that guides the user through its core functions: **Add New Student**, **View All Students**, **Search Student by Roll No**, **Update Student Record**, and **Delete Student Record**. The foundation of the system is built upon the OOP pillars: **Inheritance** allows the specialized Student class to reuse basic attributes (like name, age, phone) from the general Person class, demonstrating code efficiency. **Encapsulation** ensures data integrity by binding related data and functions together within the classes. **Polymorphism** is introduced via virtual functions, and **Function Overloading** (like in setGrade) allows a single function name to handle different data inputs. Critically, **File Handling** is implemented using C++ stream operations (ifstream and ofstream) to achieve **data persistence**, saving all student records in a pipe-separated format within the student_data.txt file, which is crucial for transactional processes like updating and deleting that require temporary file creation and renaming. This combination of OOP structure and persistent storage makes the SRMS a reliable and scalable solution for academic data management.

1. **Add New Student:** Creates and saves a new student profile.
2. **View All Students:** Displays all records currently saved.
3. **Search Student by Roll No:** Finds and displays a specific student's record.
4. **Update Student Record:** Modifies an existing student's data.
5. **Delete Student Record:** Removes a student's profile from the system.
6. **Exit:** Closes the application.

Theory of the project

OOP Concept	Where Used	Explanation
Class & Object	class Person, class Student, Student s;	The **Person** and **Student** classes define the blueprints for data. Student s; in main() creates an **Object**, which is the usable instance for all operations.
Encapsulation	All data members (name, age, roll, etc.) within classes.	**Data and the methods** that operate on that data (inputPerson(), show()) are **bundled together** inside the classes. This protects data integrity by controlling access.
Inheritance	class Student : public Person	The **Student** class **inherits** attributes (like name, phone) and methods from the **Person** base class, promoting code reuse and establishing an **"is-a"** relationship.
Polymorphism	virtual void inputPerson(), virtual void showPerson()	The use of the **virtual** keyword enables **runtime polymorphism**. This allows derived classes (Student) to override base class (Person) methods, supporting flexible design for future extensions.
Function Overloading	Two setGrade functions with different arguments.	Allows the system to have **multiple functions with the same name** (setGrade) but different parameter lists, letting the function's behavior change based on the input data type or count.
File Handling	ifstream fin, ofstream fout	Uses fstream to manage data persistence in **student_data.txt**. Operations like addRecord() (using ios::app), searchRecord(), updateRecord(), and deleteRecord() rely on file input/output streams to read, write, and manipulate the stored student data.

About the Student Record Management System

The **Student Record Management System** helps educational institutions manage student data digitally.
It eliminates manual paperwork and provides a simple way to store and retrieve information such as roll number, name, age, contact details, marks, course, branch, and teachers.

Key Features:

- Add new student details
- Display all student records
- Search student by roll number
- Update or delete student records
- Prevent duplicate entries
- Data stored permanently in a file

This system improves accuracy, saves time, and maintains organized data for academic use.

System Design:

a) Architecture:

The system follows a **modular and layered structure**:

1. **Input Layer:** Takes user input (details like name, roll number, marks, etc.)
2. **Processing Layer:** Applies validation, calculations (like grade), and stores the data.
3. **Storage Layer:** Uses file handling to save and retrieve records from student_data.txt.

4. **Output Layer:** Displays results such as student details, search results, and confirmations.

b) Class Design:

- **Class Person:** Base class for storing personal details.
- **Class Student:** Derived from Person, includes academic and course information, along with file operations.

Module Description:

Module Name	Description
1. Input Module	Collects student details such as name, roll, age, marks, course, etc.
2. File Handling Module	Saves data into student_data.txt and reads it when needed.
3. Display Module	Shows all student records or a single record (searched by roll no).
4. Search Module	Finds and displays specific student data using roll number.
5. Update Module	Updates the information of an existing student.
6. Delete Module	Deletes a record based on roll number.
7. Grade Module	Assigns grades automatically based on CGPA.
8. Menu Module	Displays main menu options and navigates user choices.

Pseudocode / Algorithm

Algorithm:

START

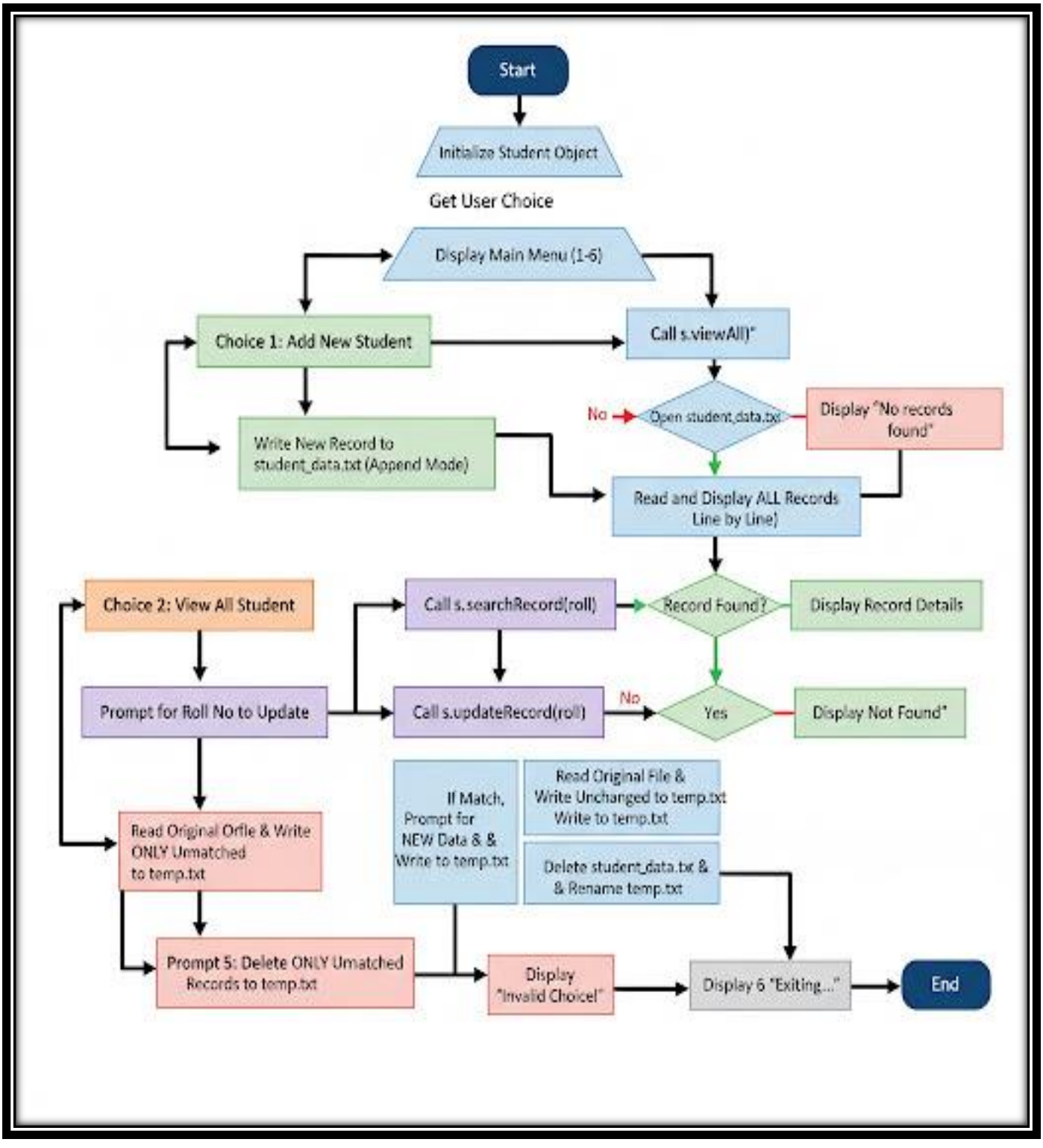
1. Display Main Menu:
    1. Add Student
    2. View All Students
    3. Search Student
    4. Update Student
    5. Delete Student
    6. Exit
2. Get user's choice.
3. IF choice = 1 THEN
    → Input student details
    → Check if roll number exists
    → Save record in file
4. IF choice = 2 THEN
    → Open file and display all records
5. IF choice = 3 THEN
    → Ask roll number
    → Search and display the record
6. IF choice = 4 THEN
    → Ask roll number
    → Update details in file
7. IF choice = 5 THEN
    → Ask roll number

→ Delete record from file

8. IF choice = 6 THEN

→ Exit the program

FLOWCHAR



SOURCE CODE

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

// ================================
// Base Class: Person
// ================================
class Person {
public:
    string name, phone, email, address;
    int age;

    void inputPerson() {
        cout << "Enter Name          : ";
        getline(cin, name);

        cout << "Enter Age           : ";
        cin >> age;
        cin.ignore();

        cout << "Enter Phone Number   : ";
```

```
        getline(cin, phone);
        cout << "Enter Email         : ";
        getline(cin, email);
        cout << "Enter Address       : ";
        getline(cin, address);
    }
    void showPerson() {
        cout << "Name       : " << name << endl;
        cout << "Age        : " << age << endl;
        cout << "Phone      : " << phone << endl;
        cout << "Email      : " << email << endl;
        cout << "Address    : " << address << endl;
    }
};

// ================================
// Derived Class: Student
// ================================
class Student : public Person {
public:
    int roll;
    float marks10, marks12, cgpa;
    string branch, course;
    char grade;
    // Grade calculator
    void setGrade() {
```

```
    if (cgpa >= 9.0) grade = 'A';
    else if (cgpa >= 8.0) grade = 'B';
    else if (cgpa >= 7.0) grade = 'C';
    else if (cgpa >= 6.0) grade = 'D';
    else grade = 'F';
}
// Input student data
void input() {
    cout << "\n---------------------------------";
    cout << "\n       ENTER STUDENT DETAILS";
    cout << "\n---------------------------------\n";

    cout << "Enter Roll No        : ";
    cin >> roll;
    cin.ignore();
    inputPerson();
    cout << "Enter 10th Marks (%) : ";
    cin >> marks10
    cout << "Enter 12th Marks (%) : ";
    cin >> marks12;
    cout << "Enter CGPA           : ";
    cin >> cgpa;
    cin.ignore();
    cout << "Enter Course         : ";
    getline(cin, course);
    cout << "Enter Branch         : ";
```

```
    getline(cin, branch);
    setGrade();
}
// Display student data
void show() {
    cout << "\n--------------------------------";
    cout << "\n       STUDENT INFORMATION";
    cout << "\n--------------------------------\n";
    cout << "Roll No    : " << roll << endl;
    showPerson();
    cout << "10th Marks : " << marks10 << "%" << endl;
    cout << "12th Marks : " << marks12 << "%" << endl;
    cout << "CGPA       : " << cgpa << endl;
    cout << "Course     : " << course << endl;
    cout << "Branch     : " << branch << endl;
    cout << "Grade      : " << grade << endl;
}
// Parse a line from file
bool parseLine(string line) {
    if (line.empty()) return false;
    stringstream ss(line);
    string token;
    int field = 0;
    try {
        while (getline(ss, token, '|')) {
            if (token.empty()) token = "0";
```

```
            switch (field) {
                case 0: roll = stoi(token); break;
                case 1: name = token; break;
                case 2: age = stoi(token); break;
                case 3: phone = token; break;
                case 4: email = token; break;
                case 5: address = token; break;
                case 6: marks10 = stof(token); break;
                case 7: marks12 = stof(token); break;
                case 8: cgpa = stof(token); break;
                case 9: course = token; break;
                case 10: branch = token; break;
                case 11: grade = token.empty() ? '-' : token[0]; break;
            }
            field++;
        }
    } catch (...) {
        return false;
    }
    return field >= 12;
}
// Check if roll number already exists
bool rollExists(int r) {
    ifstream fin("students.txt");
    if (!fin) return false;
```

```
        string line;
        while (getline(fin, line)) {
            if (line.empty()) continue
            size_t pos = line.find('|');
            if (pos != string::npos) {
                int fileRoll = stoi(line.substr(0, pos));
                if (fileRoll == r) {
                    fin.close();
                    return true;
                }
            }
        }
        fin.close();
        return false;
    }
    // Save record to file
    void saveToFile() {
        if (rollExists(roll)) {
            cout << "\nError: Roll No " << roll << " already exists!\n";
            cout << "Please use Update option to modify existing record.\n";
            return;
        }
        ofstream fout("students.txt", ios::app);
        if (fout.is_open()) {
            fout << roll << "|" << name << "|" << age << "|" << phone << "|" <<
email << "|" << address << "|"
```

```
            << marks10 << "|" << marks12 << "|" << cgpa << "|" << course <<
"|" << branch << "|" << grade << "\n";
            fout.close();
            cout << "\nRecord Saved Successfully!\n";
        } else {
            cout << "\nError saving file!\n";
        }
    }
    // View all records with proper alignment
    void viewAll() {
        ifstream fin("students.txt");
        if (!fin) {
            cout << "\nNo records found!\n";
            return;
        }
        string line;
        int count = 0;
        cout <<
"\n=============================================================
=============================================";
        cout << "\n   ALL STUDENT RECORDS";
        cout <<
"\n=============================================================
=============================================\n";
        cout << left << setw(6) << "Roll" << " | "
             << setw(20) << "Name" << " | "
             << setw(4) << "Age" << " | "
             << setw(12) << "Phone" << " | "
```

```
            << setw(25) << "Email" << " | "
            << setw(20) << "Address" << endl;
        cout << "-------------------------------------------------------------------------------
------------------------\n";
        while (getline(fin, line)) {
            if (line.empty()) continue;
            Student temp;
            if (temp.parseLine(line)) {
                // Truncate strings if they're too long
                string displayName = temp.name.length() > 20 ? temp.name.substr(0,
17) + "..." : temp.name;
                string displayEmail = temp.email.length() > 25 ? temp.email.substr(0,
22) + "..." : temp.email;
                string displayAddress = temp.address.length() > 20 ?
temp.address.substr(0, 17) + "..." : temp.address;
                cout << left << setw(6) << temp.roll << " | "
                    << setw(20) << displayName << " | "
                    << setw(4) << temp.age << " | "
                    << setw(12) << temp.phone << " | "
                    << setw(25) << displayEmail << " | "
                    << setw(20) << displayAddress << endl;
                count++;
            }
        }
        if (count == 0) {
            cout << "\nNo records found!\n";
        } else {
```

```
            cout << "-------------------------------------------------------------------------
-------------------------\n";
            cout << "Total Records: " << count << endl;
        }
        fin.close();
    }
    // Search record
    void search(int r) {
        ifstream fin("students.txt");
        if (!fin) {
            cout << "\nFile not found!\n";
            return;
        }
        string line;
        bool found = false;
        while (getline(fin, line)) {
            if (line.empty()) continue;
            size_t pos = line.find('|');
            if (pos != string::npos) {
                int fileRoll = stoi(line.substr(0, pos));
                if (fileRoll == r) {
                    Student temp;
                    if (temp.parseLine(line)) {
                        cout << "\nRecord Found!\n";
                        temp.show();
                        found = true;
```

```
                break;
            }
        }
    }
    if (!found)
        cout << "\nRecord not found!\n";
    fin.close();
}
// Delete record
void deleteRecord(int r) {
    ifstream fin("students.txt");
    if (!fin) {
        cout << "\nFile not found!\n";
        return;
    }
    ofstream temp("temp.txt");
    string line;
    bool found = false;
    while (getline(fin, line)) {
        if (line.empty()) continue;
        size_t pos = line.find('|');
        if (pos != string::npos) {
            int fileRoll = stoi(line.substr(0, pos));
            if (fileRoll != r) {
                temp << line << "\n";
```

```
            } else {
                found = true;
            }
        }
    }
    fin.close();
    temp.close();
    remove("students.txt");
    rename("temp.txt", "students.txt");
    if (found)
        cout << "\nRecord Deleted Successfully!\n";
    else
        cout << "\nRecord Not Found!\n";
}
// Update record
void updateRecord(int r) {
    ifstream fin("students.txt");
    if (!fin) {
        cout << "\nFile not found!\n";
        return;
    }
    ofstream temp("temp.txt");
    string line;
    bool found = false;
    while (getline(fin, line)) {
        if (line.empty()) continue;
```

```
            size_t pos = line.find('|');
            if (pos != string::npos) {
                int fileRoll = stoi(line.substr(0, pos));
                if (fileRoll == r) {
                    if (!found) {
                        cout << "\nCurrent Record:\n";
                        Student current;
                        if (current.parseLine(line)) {
                            current.show();
                        }
                        cout << "\nEnter New Details for Roll No " << r << ":\n";
                        input();
                        temp << roll << "|" << name << "|" << age << "|" << phone <<
"|" << email << "|" << address << "|"
                             << marks10 << "|" << marks12 << "|" << cgpa << "|" <<
course << "|" << branch << "|" << grade << "\n";
                        found = true;
                    }
                } else {
                    temp << line << "\n";
                }
            }
        }
        fin.close();
        temp.close();
        remove("students.txt");
        rename("temp.txt", "students.txt");
```

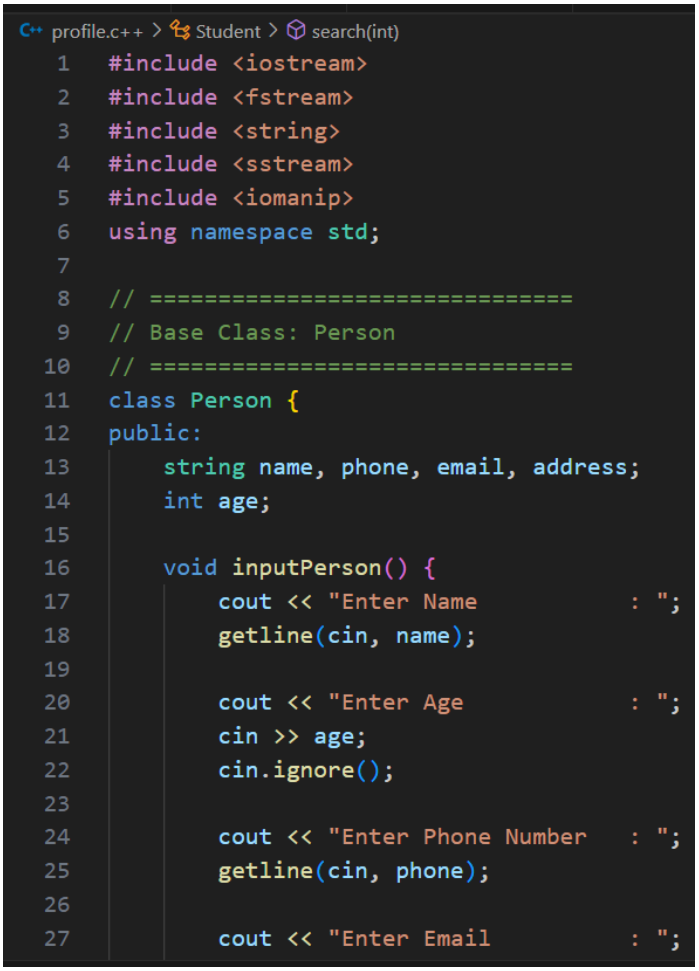
```
        if (found)
            cout << "\nRecord Updated Successfully!\n";
        else
            cout << "\nRecord Not Found!\n";
    }
};

// ================================
// Main Function
// ================================
int main() {
    Student s;
    int choice, rno;
    cout << "=======================================\n";
    cout << "  STUDENT RECORD MANAGEMENT SYSTEM  \n";
    cout << "=======================================\n";

    do {
        cout << "\n1. Add New Student";
        cout << "\n2. View All Students";
        cout << "\n3. Search Student";
        cout << "\n4. Update Student";
        cout << "\n5. Delete Student";
        cout << "\n6. Exit";
        cout << "\n---------------------------------";
        cout << "\nEnter choice: ";
```

```
if (!(cin >> choice)) {
    cin.clear();
    cin.ignore(10000, '\n');
    cout << "\nInvalid input! Please enter a number.\n";
    continue;
}
cin.ignore();
switch (choice) {
    case 1:
        s.input();
        s.saveToFile();
        break;
    case 2:
        s.viewAll();
        break;
    case 3:
        cout << "\nEnter Roll No to Search: ";
        if (cin >> rno) {
            cin.ignore();
            s.search(rno);
        } else {
            cin.clear();
            cin.ignore(10000, '\n');
            cout << "\nInvalid roll number!\n";
        }
```

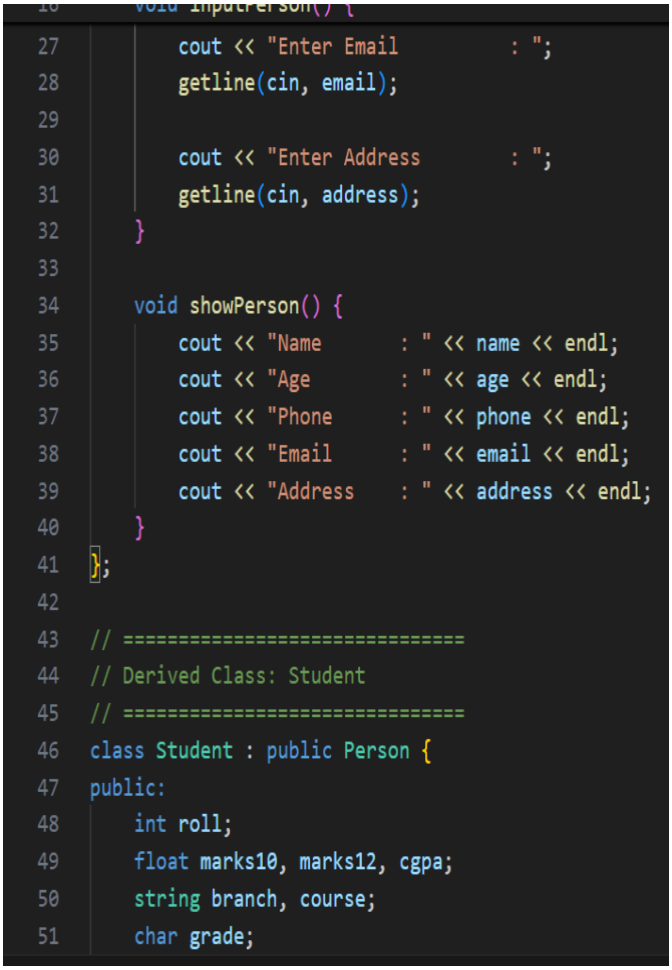
```
    break;
case 4:
    cout << "\nEnter Roll No to Update: ";
    if (cin >> rno) {
        cin.ignore();
        s.updateRecord(rno);
    } else {
        cin.clear();
        cin.ignore(10000, '\n');
        cout << "\nInvalid roll number!\n";
    }
    break;

case 5:
    cout << "\nEnter Roll No to Delete: ";
    if (cin >> rno) {
        cin.ignore();
        s.deleteRecord(rno);
    } else {
        cin.clear();
        cin.ignore(10000, '\n');
        cout << "\nInvalid roll number!\n";
    }
    break;

case 6:
```

```
            cout << "\nExiting... Thank You!\n";
            break;
        default:
            cout << "\nInvalid Choice! Try Again.\n";
    }
} while (choice != 6);
return 0;
}
```

Main code



```
profile.c++ > Student > search(int)
#include <iostream>
#include <fstream>
#include <string>
#include <sstream>
#include <iomanip>
using namespace std;

// ==============================
// Base Class: Person
// ==============================
class Person {
public:
    string name, phone, email, address;
    int age;

    void inputPerson() {
        cout << "Enter Name           : ";
        getline(cin, name);

        cout << "Enter Age            : ";
        cin >> age;
        cin.ignore();

        cout << "Enter Phone Number   : ";
        getline(cin, phone);

        cout << "Enter Email          : ";
```



```
        cout << "Enter Email          : ";
        getline(cin, email);

        cout << "Enter Address        : ";
        getline(cin, address);
    }

    void showPerson() {
        cout << "Name       : " << name << endl;
        cout << "Age        : " << age << endl;
        cout << "Phone      : " << phone << endl;
        cout << "Email      : " << email << endl;
        cout << "Address    : " << address << endl;
    }
};

// ==============================
// Derived Class: Student
// ==============================
class Student : public Person {
public:
    int roll;
    float marks10, marks12, cgpa;
    string branch, course;
    char grade;
```

```
53      // Grade calculator
54      void setGrade() {
55          if (cgpa >= 9.0) grade = 'A';
56          else if (cgpa >= 8.0) grade = 'B';
57          else if (cgpa >= 7.0) grade = 'C';
58          else if (cgpa >= 6.0) grade = 'D';
59          else grade = 'F';
60      }
61
62      // Input student data
63      void input() {
64          cout << "\n---------------------------------";
65          cout << "\n       ENTER STUDENT DETAILS";
66          cout << "\n---------------------------------\n";
67
68          cout << "Enter Roll No        : ";
69          cin >> roll;
70          cin.ignore();
71
72          inputPerson();
73
74          cout << "Enter 10th Marks (%) : ";
75          cin >> marks10;
76
77          cout << "Enter 12th Marks (%) : ";
78          cin >> marks12;
```

```
80          cout << "Enter CGPA           : ";
81          cin >> cgpa;
82          cin.ignore();
83
84          cout << "Enter Course         : ";
85          getline(cin, course);
86
87          cout << "Enter Branch         : ";
88          getline(cin, branch);
89
90          setGrade();
91      }
92
93      // Display student data
94      void show() {
95          cout << "\n---------------------------------";
96          cout << "\n       STUDENT INFORMATION";
97          cout << "\n---------------------------------\n";
98          cout << "Roll No    : " << roll << endl;
99          showPerson();
100         cout << "10th Marks : " << marks10 << "%" << endl;
101         cout << "12th Marks : " << marks12 << "%" << endl;
102         cout << "CGPA       : " << cgpa << endl;
103         cout << "Course     : " << course << endl;
104         cout << "Branch     : " << branch << endl;
```

```
105         cout << "Grade      : " << grade << endl;
106     }
107
108     // Parse a line from file
109     bool parseLine(string line) {
110         if (line.empty()) return false;
111
112         stringstream ss(line);
113         string token;
114         int field = 0;
115
116         try {
117             while (getline(ss, token, '|')) {
118                 if (token.empty()) token = "0";
119
120                 switch (field) {
121                     case 0: roll = stoi(token); break;
122                     case 1: name = token; break;
123                     case 2: age = stoi(token); break;
124                     case 3: phone = token; break;
125                     case 4: email = token; break;
126                     case 5: address = token; break;
127                     case 6: marks10 = stof(token); break;
128                     case 7: marks12 = stof(token); break;
129                     case 8: cgpa = stof(token); break;
130                     case 9: course = token; break;
131                     case 10: branch = token; break;
132                     case 11: grade = token.empty() ? '-' : token[0]; break;
133                 }
134                 field++;
135             }
136         } catch (...) {
```

```
136         } catch (...) {
137             return false;
138         }
139
140         return field >= 12;
141     }
142
143     // Check if roll number already exists
144     bool rollExists(int r) {
145         ifstream fin("students.txt");
146         if (!fin) return false;
147
148         string line;
149         while (getline(fin, line)) {
150             if (line.empty()) continue;
151
152             size_t pos = line.find('|');
153             if (pos != string::npos) {
154                 int fileRoll = stoi(line.substr(0, pos));
155                 if (fileRoll == r) {
156                     fin.close();
157                     return true;
158                 }
159             }
160         }
161         fin.close();
162         return false;
163     }
```

```
    // Save record to file
    void saveToFile() {
        if (rollExists(roll)) {
            cout << "\nError: Roll No " << roll << " already exists!\n";
            cout << "Please use Update option to modify existing record.\n";
            return;
        }

        ofstream fout("students.txt", ios::app);
        if (fout.is_open()) {
            fout << roll << "|" << name << "|" << age << "|" << phone << "|" << email << "|" << address << "|"
                 << marks10 << "|" << marks12 << "|" << cgpa << "|" << course << "|" << branch << "|" << grade << "\n";
            fout.close();
            cout << "\nRecord Saved Successfully!\n";
        } else {
            cout << "\nError saving file!\n";
        }
    }

    // View all records with proper alignment
    void viewAll() {
        ifstream fin("students.txt");
        if (!fin) {
            cout << "\nNo records found!\n";
            return;
        }

        string line;
        int count = 0;
```

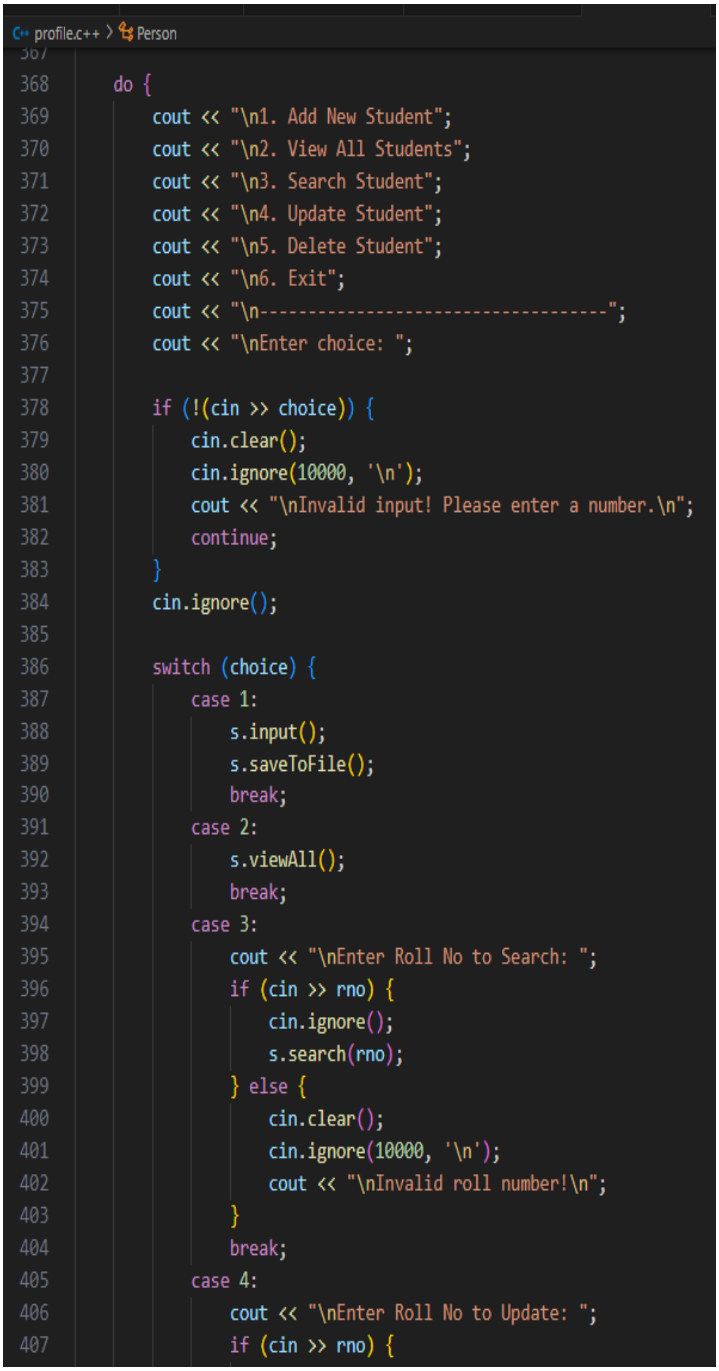
```
        cout << "\n================================================================================================================";
        cout << "\n                                        ALL STUDENT RECORDS";
        cout << "\n================================================================================================================\n";
        cout << left << setw(6) << "Roll" << " | "
             << setw(20) << "Name" << " | "
             << setw(4) << "Age" << " | "
             << setw(12) << "Phone" << " | "
             << setw(25) << "Email" << " | "
             << setw(20) << "Address" << endl;
        cout << "----------------------------------------------------------------------------------------------------------------\n";

        while (getline(fin, line)) {
            if (line.empty()) continue;

            Student temp;
            if (temp.parseLine(line)) {
                // Truncate strings if they're too long
                string displayName = temp.name.length() > 20 ? temp.name.substr(0, 17) + "..." : temp.name;
                string displayEmail = temp.email.length() > 25 ? temp.email.substr(0, 22) + "..." : temp.email;
                string displayAddress = temp.address.length() > 20 ? temp.address.substr(0, 17) + "..." : temp.address;

                cout << left << setw(6) << temp.roll << " | "
                     << setw(20) << displayName << " | "
                     << setw(4) << temp.age << " | "
                     << setw(12) << temp.phone << " | "
                     << setw(25) << displayEmail << " | "
                     << setw(20) << displayAddress << endl;
                count++;
            }
        }
```

```
        if (count == 0) {
            cout << "\nNo records found!\n";
        } else {
            cout << "----------------------------------------------------------------------------------------------------------------\n";
            cout << "Total Records: " << count << endl;
        }

        fin.close();
    }

    // Search record
    void search(int r) {
        ifstream fin("students.txt");
        if (!fin) {
            cout << "\nFile not found!\n";
            return;
        }

        string line;
        bool found = false;
        while (getline(fin, line)) {
            if (line.empty()) continue;

            size_t pos = line.find('|');
            if (pos != string::npos) {
                int fileRoll = stoi(line.substr(0, pos));
                if (fileRoll == r) {
                    Student temp;
                    if (temp.parseLine(line)) {
                        cout << "\nRecord Found!\n";
                        temp.show();
```

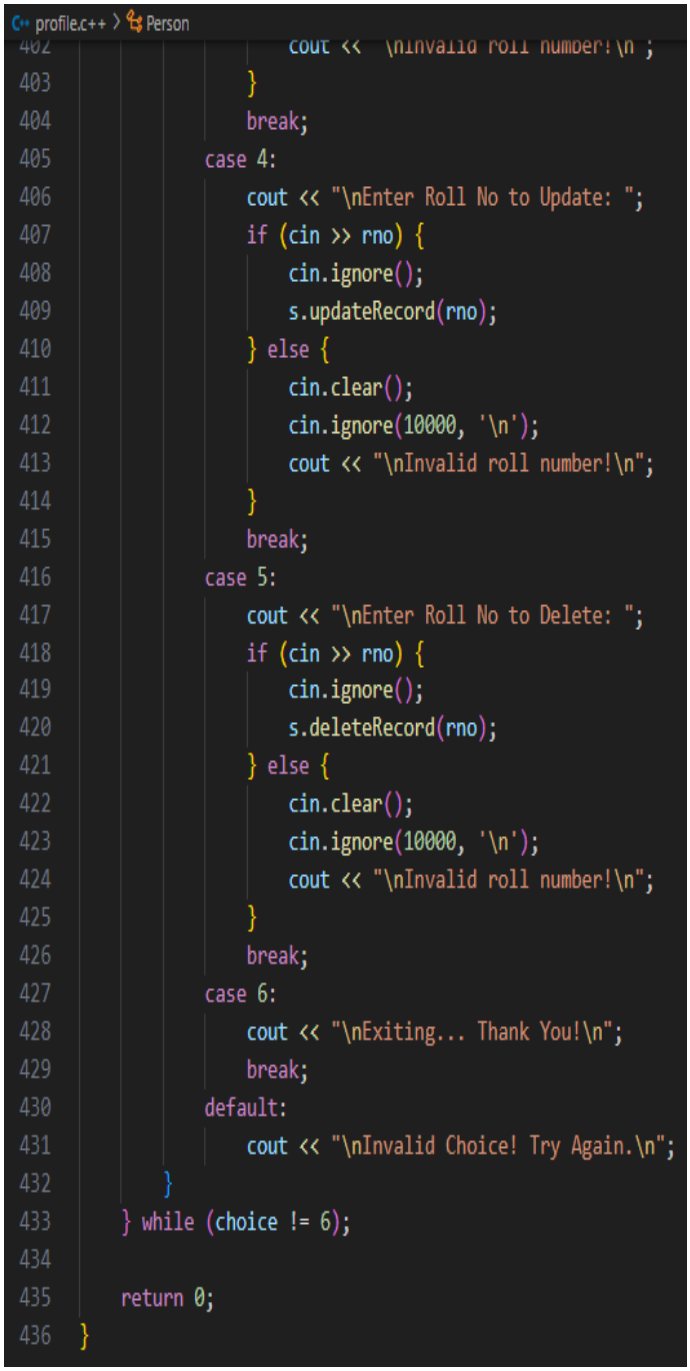
```
                    found = true;
                    break;
                }
            }
        }
    }
    if (!found)
        cout << "\nRecord not found!\n";
    fin.close();
}

// Delete record
void deleteRecord(int r) {
    ifstream fin("students.txt");
    if (!fin) {
        cout << "\nFile not found!\n";
        return;
    }

    ofstream temp("temp.txt");
    string line;
    bool found = false;

    while (getline(fin, line)) {
        if (line.empty()) continue;

        size_t pos = line.find('|');
        if (pos != string::npos) {
            int fileRoll = stoi(line.substr(0, pos));
            if (fileRoll != r) {
                temp << line << "\n";
            } else {
                found = true;
            }
        }
    }
```

```

    fin.close();
    temp.close();

    remove("students.txt");
    rename("temp.txt", "students.txt");

    if (found)
        cout << "\nRecord Deleted Successfully!\n";
    else
        cout << "\nRecord Not Found!\n";
}

// Update record
void updateRecord(int r) {
    ifstream fin("students.txt");
    if (!fin) {
        cout << "\nFile not found!\n";
        return;
    }

    ofstream temp("temp.txt");
    string line;
    bool found = false;

    while (getline(fin, line)) {
        if (line.empty()) continue;

        size_t pos = line.find('|');
        if (pos != string::npos) {
            int fileRoll = stoi(line.substr(0, pos));
            if (fileRoll == r) {
                if (!found) {
                    cout << "\nCurrent Record:\n";
                    Student current;
                    if (current.parseLine(line)) {
                        current.show();
                    }
```

```
                    cout << "\nEnter New Details for Roll No " << r << ":\n";
                    input();
                    temp << roll << "|" << name << "|" << age << "|" << phone << "|" << email << "|" << address << "|"
                         << marks10 << "|" << marks12 << "|" << cgpa << "|" << course << "|" << branch << "|" << grade << "\n";
                    found = true;
                }
            } else {
                temp << line << "\n";
            }
        }
    }

    fin.close();
    temp.close();

    remove("students.txt");
    rename("temp.txt", "students.txt");

    if (found)
        cout << "\nRecord Updated Successfully!\n";
    else
        cout << "\nRecord Not Found!\n";
}
};

// ==============================
// Main Function
// ==============================
int main() {
    Student s;
    int choice, rno;

    cout << "==================================\n";
    cout << "  STUDENT RECORD MANAGEMENT SYSTEM  \n";
    cout << "==================================\n";
```



```
profile.c++ > Person
367
368     do {
369         cout << "\n1. Add New Student";
370         cout << "\n2. View All Students";
371         cout << "\n3. Search Student";
372         cout << "\n4. Update Student";
373         cout << "\n5. Delete Student";
374         cout << "\n6. Exit";
375         cout << "\n----------------------------------";
376         cout << "\nEnter choice: ";
377
378         if (!(cin >> choice)) {
379             cin.clear();
380             cin.ignore(10000, '\n');
381             cout << "\nInvalid input! Please enter a number.\n";
382             continue;
383         }
384         cin.ignore();
385
386         switch (choice) {
387             case 1:
388                 s.input();
389                 s.saveToFile();
390                 break;
391             case 2:
392                 s.viewAll();
393                 break;
394             case 3:
395                 cout << "\nEnter Roll No to Search: ";
396                 if (cin >> rno) {
397                     cin.ignore();
398                     s.search(rno);
399                 } else {
400                     cin.clear();
401                     cin.ignore(10000, '\n');
402                     cout << "\nInvalid roll number!\n";
403                 }
404                 break;
405             case 4:
406                 cout << "\nEnter Roll No to Update: ";
407                 if (cin >> rno) {
```



```
profile.c++ > Person
402                     cout << "\nInvalid roll number!\n";
403                 }
404                 break;
405             case 4:
406                 cout << "\nEnter Roll No to Update: ";
407                 if (cin >> rno) {
408                     cin.ignore();
409                     s.updateRecord(rno);
410                 } else {
411                     cin.clear();
412                     cin.ignore(10000, '\n');
413                     cout << "\nInvalid roll number!\n";
414                 }
415                 break;
416             case 5:
417                 cout << "\nEnter Roll No to Delete: ";
418                 if (cin >> rno) {
419                     cin.ignore();
420                     s.deleteRecord(rno);
421                 } else {
422                     cin.clear();
423                     cin.ignore(10000, '\n');
424                     cout << "\nInvalid roll number!\n";
425                 }
426                 break;
427             case 6:
428                 cout << "\nExiting... Thank You!\n";
429                 break;
430             default:
431                 cout << "\nInvalid Choice! Try Again.\n";
432         }
433     } while (choice != 6);
434
435     return 0;
436 }
```

Output

```
==================================
  STUDENT RECORD MANAGEMENT SYSTEM
==================================

1. Add New Student
2. View All Students
3. Search Student
4. Update Student
5. Delete Student
6. Exit
----------------------------------
Enter choice: 2

No records found!

1. Add New Student
2. View All Students
3. Search Student
4. Update Student
5. Delete Student
6. Exit
----------------------------------
Enter choice: 34

Invalid Choice! Try Again.

1. Add New Student
2. View All Students
3. Search Student
4. Update Student
5. Delete Student
6. Exit
----------------------------------
Enter choice: 21

Invalid Choice! Try Again.
```

```
1. Add New Student
2. View All Students
3. Search Student
4. Update Student
5. Delete Student
6. Exit
----------------------------------
Enter choice: 1

---------------------------------
       ENTER STUDENT DETAILS
---------------------------------
Enter Roll No         : 12
Enter Name            : arushi saxena
Enter Age             : 20
Enter Phone Number    : 2345678631
Enter Email           : arushisaxena@1313gmail.com
Enter Address         : lucknow
Enter 10th Marks (%) : 80
Enter 12th Marks (%) : 86
Enter CGPA            : 8.55
Enter Course          : bca
Enter Branch          : uic

Record Saved Successfully!
```

```
1. Add New Student
2. View All Students
3. Search Student
4. Update Student
5. Delete Student
6. Exit
----------------------------------
Enter choice: 1

---------------------------------
       ENTER STUDENT DETAILS
---------------------------------
Enter Roll No         : 30
Enter Name            : ananya singh
Enter Age             : 21
Enter Phone Number    : 3475941269
Enter Email           : anusingh@0925gmail.com
Enter Address         : kanpur
Enter 10th Marks (%) : 90
Enter 12th Marks (%) : 95
Enter CGPA            : 9.1
Enter Course          : btech
Enter Branch          : engineering

Record Saved Successfully!
```

```
1. Add New Student
2. View All Students
3. Search Student
4. Update Student
5. Delete Student
6. Exit
----------------------------------
Enter choice: 2

=================================================================================================
                                        ALL STUDENT RECORDS
=================================================================================================
Roll   | Name                | Age  | Phone        | Email                    | Address
-------------------------------------------------------------------------------------------------
12     | arushi saxena       | 20   | 2345678631   | arushisaxena@1313gmail... | lucknow
30     | ananya singh        | 21   | 3475941269   | anusingh@0925gmail.com    | kanpur
-------------------------------------------------------------------------------------------------
Total Records: 2
```

```
1. Add New Student
2. View All Students
3. Search Student
4. Update Student
5. Delete Student
6. Exit
----------------------------------
Enter choice: 3

Enter Roll No to Search: 12

Record Found!

----------------------------------
       STUDENT INFORMATION
----------------------------------
Roll No     : 12
Name        : arushi saxena
Age         : 20
Phone       : 2345678631
Email       : arushisaxena@1313gmail.com
Address     : lucknow
10th Marks  : 80%
12th Marks  : 86%
CGPA        : 8.55
Course      : bca
Branch      : uic
Grade       : B
```

```
1. Add New Student
2. View All Students
3. Search Student
4. Update Student
5. Delete Student
6. Exit
----------------------------------
Enter choice: 4

Enter Roll No to Update: 23

Record Not Found!

1. Add New Student
2. View All Students
3. Search Student
4. Update Student
5. Delete Student
6. Exit
----------------------------------
Enter choice: 4

Enter Roll No to Update: 12
```

```
Current Record:

---------------------------------
       STUDENT INFORMATION
---------------------------------
Roll No     : 12
Name        : arushi saxena
Age         : 20
Phone       : 2345678631
Email       : arushisaxena@1313gmail.com
Address     : lucknow
10th Marks : 80%
12th Marks : 86%
CGPA        : 8.55
Course      : bca
Branch      : uic
Grade       : B

Enter New Details for Roll No 12:

---------------------------------
       ENTER STUDENT DETAILS
---------------------------------
Enter Roll No         : 25
Enter Name            : chitransh saxena
Enter Age             : 23
Enter Phone Number    : 5678453456
Enter Email           : chitranshsaxena@1414gmail.com
Enter Address         : delhi
Enter 10th Marks (%) : 80
Enter 12th Marks (%) : 90
Enter CGPA            : 9.2
Enter Course          : bcom
Enter Branch          : commernce

Record Updated Successfully!
```

```
1. Add New Student
2. View All Students
3. Search Student
4. Update Student
5. Delete Student
6. Exit
-------------------------------------
Enter choice: 5

Enter Roll No to Delete: 78

Record Not Found!

1. Add New Student
2. View All Students
3. Search Student
4. Update Student
5. Delete Student
6. Exit
-------------------------------------
Enter choice: 5

Enter Roll No to Delete: 25

Record Deleted Successfully!

1. Add New Student
-------------------------------------
Enter choice: 5

Enter Roll No to Delete: 25

Record Deleted Successfully!

1. Add New Student
Enter Roll No to Delete: 25

Record Deleted Successfully!
```

```
1. Add New Student

1. Add New Student
1. Add New Student
2. View All Students
3. Search Student
4. Update Student
5. Delete Student
6. Exit
-------------------------------------
Enter choice: 6

4. Update Student
5. Delete Student
6. Exit
-------------------------------------
Enter choice: 6

5. Delete Student
6. Exit
-------------------------------------
Enter choice: 6

6. Exit
-------------------------------------
Enter choice: 6

-------------------------------------
Enter choice: 6

Enter choice: 6


Exiting... Thank You!

c:\Users\arush\OneDrive\Desktop\c++ codes>
```

CONCLUSION

In conclusion, the **Student Record Management System (SRMS)** project has effectively illustrated essential programming concepts, including **data modelling using OOP**, file handling for **data persistence**, and **menu-driven user interaction**. By implementing core functionalities like adding new students, searching, updating, and deleting records, we created a practical and vital tool for administrative data management. Throughout the development process, we encountered challenges such as ensuring the correct **parsing of pipe-separated file data**, maintaining **data integrity** during update and delete operations, and successfully applying **Object-Oriented Programming (OOP) principles** like Inheritance. Overcoming these obstacles enhanced our problem-solving skills and deepened our understanding of **C++ classes, streaming, and file manipulation**.

The project also emphasized the significance of **code organization and maintainability**, clearly segmenting logic into classes (Person, Student) and dedicated methods (addRecord, searchRecord). This structure allows for easy updates and feature additions. Future iterations could explore enhancements like a graphical user interface (GUI), database integration (instead of flat files) for better security and scalability, or adding complex reporting features like merit list generation. Overall, this Student Record Management System project has not only solidified our programming skills but has also served as a foundation for further exploration into **data structures and enterprise application development**. It has been a rewarding experience that highlights the intersection of logic, structure, and technology in building essential administrative tools.

BIBLIOGRAPHY

- ✓ Chatgpt
- ✓ Wikipedia
- ✓ Geeksforgeeks
- ✓ Scribd
- ✓ Freecodecamp forum
- ✓ React.dev

THANK YOU……