Practical Exercise

Student Record System Implementation Using Structure

The **objective** of this exercise is to build a practical **Student Record Management System** using **structure** in C++. It helps to apply the structure concepts covered in lecture #5.

The requirements are as follows

- The structure should be able to store up to 100 student records.
- The expected fields for each student are:
    - Student ID — an integer,
    - Name — a string,
    - GPA — a float, and
    - Department — a string.

The features the system should support include:

- Add student — to insert a new record,
- List all students — to display all records, and
- Search by ID — to find a specific student.

Solution: Student Management System (Structures + Functions)

student.cpp

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

using namespace std;

const int MAX_STUDENTS = 100;

// structure declaration/definition
struct Student {
    int id;
    string name;
    float gpa;
    string department;
};

// function prototype declaration
void addStudent(Student s[], int &count);
void listStudents(Student s[], int count);
int findById(Student s[], int count, int id);
void searchStudent(Student s[], int count);
```

```
int main() {
    Student students[MAX_STUDENTS];
    int count = 0, choice;

    while(true) {
        cout << "\n1. Add 2. List 3. Search 4. Exit: ";
        cin >> choice;

        switch(choice) {
            case 1:
                addStudent(students, count);
                break;
            case 2:
                listStudents(students, count);
                break;
            case 3:
                searchStudent(students, count);
                break;
            case 4:
               return 0;
            default:
                cout << "Invalid choice!\n";
        }
    }
    return 0;
}
```

```cpp
void addStudent(Student s[], int& count) {

    if(count >= MAX_STUDENTS) {
        cout << "Student limit reached!" << endl;
        return;
    }

    cout << "Enter ID: ";
    cin >> s[count].id;

    cout << "Enter Name: ";
    cin >> s[count].name;

    cout << "Enter GPA: ";
    cin >> s[count].gpa;

    cout << "Enter Department: ";
    cin >> s[count].department;

    count++;

    cout << "Student added successfully!\n";
}
```

```cpp
void listStudents(Student s[], int count) {

    if(count == 0) {
        cout << "No records found.\n";
        return;
    }

    cout << "\n--- Student List ---\n";

    for(int i = 0; i < count; i++) {
        cout << i + 1<< " "
        << s[i].name << " ("<< s[i].id << "), GPA: "
        << fixed << setprecision(2)<< s[i].gpa
        << ", Dept: "<< s[i].department << ")"<< endl;
    }
}
```

```
void searchStudent(Student s[], int count) {

    int id;

    cout << "Enter ID to search: ";
    cin >> id;

    int index = findById(s, count, id);

    if(index == -1) {

        cout << "Student not found.\n";

    } else {

        cout << "\nStudent Found:\n";

        cout << "Name: "<< s[index].name << endl;
        cout << "ID: " << s[index].id << endl;
        cout << "GPA: "<< s[index].gpa << endl;
        cout << "Department: " << s[index].department<< endl;
    }
}
```

```
int findById(Student s[],
     int count,
     int id) {

    for(int i = 0; i < count; i++) {

        if(s[i].id == id)
            return i;
    }

    return -1;
}
```

Program Output

```
Menu:
1. Add, 2. List, 3. Search, 4. Exit:
Your Choice: 1

Adding New Students
Enter ID: 101
Enter Name: Abel
Enter GPA: 3.75
Enter Department: Software

Student added successfully!
Menu:
1. Add, 2. List, 3. Search, 4. Exit:
Your Choice: 1

Adding New Students
Enter ID: 102
Enter Name: Hana
Enter GPA: 3.90
Enter Department: Electrical

Student added successfully!

Menu:
1. Add, 2. List, 3. Search, 4. Exit:
Your Choice: 2

--- Student List ---

1. Abel (ID: 101, GPA: 3.75, Dept: Software)
2. Hana (ID: 102, GPA: 3.90, Dept: Electrical)
```

```
Menu:
1. Add, 2. List, 3. Search, 4. Exit:
Your Choice: 3

Enter ID to search: 102

Student Found:
Name: Hana
ID: 102
GPA: 3.9
Department: Electrical

Menu:
1. Add, 2. List, 3. Search, 4. Exit:
Your Choice: 4
```