```
#include <iostream>
#include <cstring>
#include <cstdlib> // For malloc, realloc, free
using namespace std;

struct Student {
    int id;
    char name[50];
    float cgpa;
};

void addStudent(Student*& students, int& size, int& capacity) {
    if (size == capacity) {
        capacity *= 2;
        students = (Student*)realloc(students, capacity * sizeof(Student));
        if (!students) {
            cout << "Memory allocation failed!" << endl;
            exit(1);
        }
    }
    cout << "Enter Student ID: ";
    cin >> students[size].id;
    cout << "Enter Student Name: ";
    cin.ignore();
    cin.getline(students[size].name, 50);
    cout << "Enter Student CGPA: ";
    cin >> students[size].cgpa;
    size++;
}
```

```
void linearSearch(Student* students, int size, int id) {
    for (int i = 0; i < size; i++) {
        if (students[i].id == id) {
            cout << "Student Found: " << students[i].name << ", CGPA: " << students[i].cgpa << endl;
            return;
        }
    }
    cout << "Student not found!" << endl;
}

void binarySearch(Student* students, int size, int id) {
    int low = 0, high = size - 1;
    while (low <= high) {
        int mid = low + (high - low) / 2;
        if (students[mid].id == id) {
            cout << "Student Found: " << students[mid].name << ", CGPA: " << students[mid].cgpa <<
endl;
            return;
        } else if (students[mid].id < id) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }
    cout << "Student not found!" << endl;
}

void bubbleSortByName(Student* students, int size) {
    for (int i = 0; i < size - 1; i++) {
        for (int j = 0; j < size - i - 1; j++) {
            if (strcmp(students[j].name, students[j + 1].name) > 0) {
```

```
            swap(students[j], students[j + 1]);
        }
    }
}

void selectionSortByCGPA(Student* students, int size, bool ascending = true) {
    for (int i = 0; i < size - 1; i++) {
        int minIndex = i;
        for (int j = i + 1; j < size; j++) {
            if (ascending ? (students[j].cgpa < students[minIndex].cgpa) : (students[j].cgpa > students[minIndex].cgpa)) {
                minIndex = j;
            }
        }
        swap(students[i], students[minIndex]);
    }
}

void displayStudents(Student* students, int size) {
    for (int i = 0; i < size; i++) {
        cout << "ID: " << students[i].id << ", Name: " << students[i].name << ", CGPA: " << students[i].cgpa << endl;
    }
}

int main() {
    int capacity = 2, size = 0;
    Student* students = (Student*)malloc(capacity * sizeof(Student));
    if (!students) {
        cout << "Memory allocation failed!" << endl;
        return 1;
```

```
}

int choice;
do {
    cout << "\n===== STUDENT DATABASE MENU =====\n";
    cout << "1. Add Student\n";
    cout << "2. Display Students\n";
    cout << "3. Linear Search by ID\n";
    cout << "4. Binary Search by ID\n";
    cout << "5. Bubble Sort by Name (Alphabetically)\n";
    cout << "6. Selection Sort by CGPA (Ascending Order)\n";
    cout << "7. Selection Sort by CGPA (Descending Order)\n";
    cout << "8. Exit\n";
    cout << "Enter your choice: ";
    cin >> choice;

    switch (choice) {
        case 1:
            addStudent(students, size, capacity);
            break;
        case 2:
            displayStudents(students, size);
            break;
        case 3: {
            int id;
            cout << "Enter Student ID to search: ";
            cin >> id;
            linearSearch(students, size, id);
            break;
        }
        case 4: {
```

```
            int id;
            cout << "Enter Student ID to search: ";
            cin >> id;
            binarySearch(students, size, id);
            break;
        }
        case 5:
            bubbleSortByName(students, size);
            cout << "Sorted by Name (Alphabetically).\n";
            break;
        case 6:
            selectionSortByCGPA(students, size, true);
            cout << "Sorted by CGPA (Ascending).\n";
            break;
        case 7:
            selectionSortByCGPA(students, size, false);
            cout << "Sorted by CGPA (Descending).\n";
            break;
        case 8:
            cout << "Exiting program.\n";
            break;
        default:
            cout << "Invalid choice! Try again.\n";
      }
   } while (choice != 8);

   free(students); // Free allocated memory
   return 0;
}
```