Solve the Assignment using C++

Consider a student database of SEIT class (at least 15 records).
Database contains different fields of every student like Roll No, Name and SGPA. (array of structure)

a) Sorting

1) Design a roll call list, arrange list of students according to roll numbers in order (Use Bubble Sort)

2 Arrange list of students alphabetically. (Use Insertion sort)

3) Arrange list of students to find out first ten toppers from a class. (Use Quick sort) (V lab)

b) Searching

1) Search students according to SGPA. If more than one student having same SGPA, then print list of all students having same SGPA.

2) Search a particular student according to name using binary search without recursion.

(all the student records having the presence of search key should be displayed)

Part a) Sorting

1. **Bubble Sort** for Roll Numbers
2. **Insertion Sort** for Names
3. **Quick Sort** for SGPA to get top 10

Part b) Searching

1. **Linear Search by SGPA**
2. **Binary Search by Name** (non-recursive)

Complete C++ Program

```
#include <iostream.h>
#include<coni.h>

#include <string>
using namespace std;

struct Student {
    int rollNo;
    string name;
    float sgpa;
};

// 1) Bubble Sort by Roll Number
void bubbleSortByRoll(Student s[], int n) {
    for (int i = 0; i < n-1; i++) {
        for (int j = 0; j < n-i-1; j++) {
            if (s[j].rollNo > s[j+1].rollNo) {
                swap(s[j], s[j+1]);
            }
        }
    }
}

// 2) Insertion Sort by Name
void insertionSortByName(Student s[], int n) {
    for (int i = 1; i < n; i++) {
        Student key = s[i];
        int j = i - 1;
        while (j >= 0 && s[j].name > key.name) {
            s[j+1] = s[j];
            j--;
        }
        s[j+1] = key;
    }
}

// 3) Quick Sort by SGPA (Descending)
int partition(Student s[], int low, int high) {
    float pivot = s[high].sgpa;
    int i = low - 1;

    for (int j = low; j < high; j++) {
        if (s[j].sgpa > pivot) {  // Descending
            i++;
            swap(s[i], s[j]);
        }
    }
    swap(s[i+1], s[high]);
    return i+1;
}

void quickSortBySGPA(Student s[], int low, int high) {
    if (low < high) {
        int pi = partition(s, low, high);
        quickSortBySGPA(s, low, pi - 1);
        quickSortBySGPA(s, pi + 1, high);
    }
}

// Display student list
void display(Student s[], int n) {
    cout << "\nRoll No\tName\tSGPA\n";
    cout << "----------------------------\n";
    for (int i = 0; i < n; i++) {
        cout << s[i].rollNo << "\t" << s[i].name << "\t" << s[i].sgpa << endl;
    }
}

int main() {
    int n = 15;
    Student s[15] = {
        {105, "Amit", 8.1}, {101, "Rita", 9.2}, {110, "Karan", 6.8},
        {103, "Divya", 7.5}, {108, "Neha", 9.0}, {102, "Bhavesh", 8.6},
        {104, "Manish", 5.9}, {106, "Sneha", 8.9}, {109, "Pooja", 7.8},
        {107, "Yash", 9.5}, {112, "Omkar", 6.5}, {111, "Tina", 7.0},
        {113, "Umesh", 6.2}, {114, "Geeta", 8.7}, {115, "Rohan", 9.1}
    };

    cout << "Original List:";
    display(s, n);

    // 1) Bubble Sort by Roll No
    bubbleSortByRoll(s, n);
    cout << "\nSorted by Roll Number (Bubble Sort):";
    display(s, n);

    // 2) Insertion Sort by Name
    insertionSortByName(s, n);
    cout << "\nSorted Alphabetically by Name (Insertion Sort):";
    display(s, n);

    // 3) Quick Sort by SGPA
    quickSortBySGPA(s, 0, n-1);
    cout << "\nTop 10 Students by SGPA (Quick Sort):";
    display(s, 10);  // Only top 10
    return 0;
}
```