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

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

// Function to display all students
void showAll(Student* students[], int n)
{
   for (int i = 0; i < n; i++)
        {
      cout << "Student Name: " << students[i]->name << ", Roll No: "
<< students[i]->roll << ", SGPA: " << students[i]->sgpa << "\n";
   }
}

// Function to display the top 10 students
void showTop10(Student* students[], int n)
{
   int limit = (n < 10) ? n : 10;
   for (int i = n - 1; i >= n - limit; i--)
        {
      cout << "Student Name: " << students[i]->name << ", Roll No: "<<
students[i]->roll << ", SGPA: " << students[i]->sgpa << "\n";
   }
}

// Bubble sort by roll number
void bubbleSort(Student* students[], int n)
{
   for (int i = 0; i < n - 1; i++)
        {
      for (int j = 0; j < n - i - 1; j++)
```

```
                {
            if (students[j]->roll > students[j + 1]->roll)
                        {
                swap(students[j], students[j + 1]);
            }
        }
    }
}

// Insertion sort by name
void insertionSort(Student* students[], int n)
{
    for (int i = 1; i < n; i++)
          {
        Student* key = students[i];
        int j = i - 1;
        while (j >= 0 && students[j]->name > key->name)
                {
            students[j + 1] = students[j];
            j--;
        }
        students[j + 1] = key;
    }
}

// Partition function for QuickSort (by SGPA)
int partition(Student* students[], int low, int high)
{
    float pivot = students[low]->sgpa;
    int start = low, end = high;
    while (start < end)
          {
        while (students[start]->sgpa <= pivot && start < high)
                {
            start++;
        }
        while (students[end]->sgpa > pivot)
```

```
                {
            end--;
        }
        if (start < end)
                {
            swap(students[start], students[end]);
        }
    }
    swap(students[low], students[end]);
    return end;
}

// QuickSort by SGPA
void quickSort(Student* students[], int low, int high) {
    if (low < high) {
        int pivot = partition(students, low, high);
        quickSort(students, low, pivot - 1);
        quickSort(students, pivot + 1, high);
    }
}

// Linear search by SGPA
void linearSearch(Student* students[], int n, float sgpa)
{
    bool found = false;
    for (int i = 0; i < n; i++)
          {
        if (students[i]->sgpa == sgpa)
                {
            cout << "Student Name: " << students[i]->name << ", Roll No: "<<
students[i]->roll << ", SGPA: " << students[i]->sgpa << "\n";
            found = true;
        }
    }
    if (!found)
          {
        cout << "NO MATCH FOUND\n";
    }
}

// Binary search by name (assumes sorted by name)
void binarySearch(Student* students[], int n, const string& name)
{
    int low = 0, high = n - 1;
    bool found = false;
    while (low <= high)
        {
        int mid = (low + high) / 2;
        if (students[mid]->name == name)
                {
            found = true;
            // Display all matches
            int i = mid;
            while (i >= 0 && students[i]->name == name) i--;
            i++;
            while (i < n && students[i]->name == name)
                        {
                cout << "Student Name: " << students[i]->name << ", Roll No: "
                    << students[i]->roll << ", SGPA: " << students[i]->sgpa <<
"\n";
                i++;
            }
            break;
        } else if (students[mid]->name < name)
                {
            low = mid + 1;
        } else
                {
            high = mid - 1;
        }
    }
    if (!found)
        {
        cout << "NO MATCH FOUND\n";
    }
}

// Main function
int main()
{
    Student* students[60];
    int n = 0, choice;
    char cont;

    do {
        cout << "\nMENU:\n";
        cout << "1. Insert Records\n";
        cout << "2. Display Class Details (Sorted by Roll Number)\n";
        cout << "3. Display Top 10 Students (Sorted by SGPA)\n";
        cout << "4. Display Class Details (Sorted by Name)\n";
        cout << "5. Find Student by SGPA\n";
        cout << "6. Find Student by Name\n";
        cout << "Enter your choice: ";
        cin >> choice;

        switch (choice)
                {
            case 1:
                        {
                int addCount;
                cout << "How many students do you want to add? ";
                cin >> addCount;
                for (int i = 0; i < addCount; i++)
                            {
                    students[n] = new Student;
                    cout << "Enter Roll Number: ";
                    cin >> students[n]->roll;
                    cout << "Enter Name: ";
                    cin >> students[n]->name;
                    cout << "Enter SGPA: ";
                    cin >> students[n]->sgpa;
                    n++;
                }
                break;
            }
            case 2:
                bubbleSort(students, n);
                cout << "Class Details (Sorted by Roll Number):\n";
                showAll(students, n);
                break;
            case 3:
                quickSort(students, 0, n - 1);
                cout << "Top 10 Students:\n";
                showTop10(students, n);
                break;
            case 4:
                insertionSort(students, n);
                cout << "Class Details (Sorted by Name):\n";
                showAll(students, n);
                break;
            case 5:
                        {
                float sgpa;
                cout << "Enter SGPA to search: ";
                cin >> sgpa;
                linearSearch(students, n, sgpa);
                break;
            }
            case 6:
                        {
                string name;
                cout << "Enter Name to search: ";
                cin >> name;
                insertionSort(students, n); // Ensure the list is sorted by name
                binarySearch(students, n, name);
                break;
            }
            default:
```

```
                cout << "Invalid choice! Try again.\n";
            }

            cout << "Do you want to continue? (y/n): ";
            cin >> cont;

        } while (cont == 'y' || cont == 'Y');

        // Clean up dynamically allocated memory
        for (int i = 0; i < n; i++)
              {
            delete students[i];
        }

        return 0;
    }
```