#include <iostream>

using namespace std;

struct Student

{

int Roll\_No;

string Name;

float SGPA;

};

void quicksort(Student students[], int low, int high);

int partition(Student students[], int low, int high);

int main()

{

cout << "Enter number of students to accept: ";

int n;

cin >> n;

Student students[n];

// Input

for (int i = 0; i < n; i++)

{

cout << "Enter Roll No of Student: ";

cin >> students[i].Roll\_No;

cout << "Enter Name of Student: ";

cin >> students[i].Name;

cout << "Enter SGPA of Student: ";

cin >> students[i].SGPA;

cout << endl;

}

// Bubble sort by Roll\_No

for (int i = 0; i < n - 1; i++)

{

for (int j = 0; j < n - 1 - i; j++)

{

if (students[j].Roll\_No > students[j + 1].Roll\_No)

{

Student temp = students[j];

students[j] = students[j + 1];

students[j + 1] = temp;

}

}

}

// Output Roll Number-wise order

cout << endl << "Roll Number-wise order" << endl;

for (int i = 0; i < n; i++)

{

cout << "Student " << i + 1 << " details:" << endl;

cout << "Roll No: " << students[i].Roll\_No << ", Name: " << students[i].Name << ", SGPA: " << students[i].SGPA << endl;

}

// Insertion sort by Name

for (int i = 1; i < n; i++)

{

Student tosort = students[i];

int j = i - 1;

while (j >= 0 && students[j].Name.compare(tosort.Name) > 0)

{

students[j + 1] = students[j];

j = j - 1;

}

students[j + 1] = tosort;

}

// Output Alphabetical order

cout << endl << "Alphabetical order" << endl;

for (int i = 0; i < n; i++)

{

cout << "Student " << i + 1 << " details:" << endl;

cout << "Roll No: " << students[i].Roll\_No << ", Name: " << students[i].Name << ", SGPA: " << students[i].SGPA << endl;

}

// Quicksort by SGPA

quicksort(students, 0, n - 1);

// Output SGPA order

cout << endl << "SGPA order" << endl;

for (int i = n - 1; i >= 0; i--)

{

cout << "Student details:" << endl;

cout << "Roll No: " << students[i].Roll\_No << ", Name: " << students[i].Name << ", SGPA: " << students[i].SGPA << endl;

}

// Linear search

cout << endl << "Enter SGPA to search (Linear): ";

float searchl;

cin >> searchl;

int num = 0;

for (int i = 0; i < n; i++)

{

if (students[i].SGPA == searchl)

{

cout << "Roll No: " << students[i].Roll\_No << ", Name: " << students[i].Name << ", SGPA: " << students[i].SGPA << endl;

num++;

}

}

if (num == 0)

{

cout << "No Students with given SGPA found";

}

// Binary search

cout << endl << "Enter SGPA to search (Binary): ";

float searchb;

cin >> searchb;

int high = n - 1;

int low = 0;

int found = 0;

while (low <= high)

{

int mid = (high + low) / 2;

if (students[mid].SGPA == searchb)

{

cout << "Roll No: " << students[mid].Roll\_No << ", Name: " << students[mid].Name << ", SGPA: " << students[mid].SGPA << endl;

found++;

break;

}

else if (students[mid].SGPA < searchb)

{

low = mid + 1;

}

else

{

high = mid - 1;

}

}

if (found == 0)

{

cout << "No Students with given SGPA found" << endl;

}

}

// Quicksort functions

int partition(Student students[], int low, int high)

{

Student pivot = students[high];

int i = low - 1;

for (int j = low; j < high; j++)

{

if (students[j].SGPA < pivot.SGPA)

{

i++;

Student temp1 = students[i];

students[i] = students[j];

students[j] = temp1;

}

}

Student temp2 = students[i + 1];

students[i + 1] = students[high];

students[high] = temp2;

return i + 1;

}

void quicksort(Student students[], int low, int high)

{

if (low < high)

{

int pivot\_index = partition(students, low, high);

quicksort(students, low, pivot\_index - 1);

quicksort(students, pivot\_index + 1, high);

}

}