#include <iostream>

#include <string>

using namespace std;

struct student {

string name;

int rollno;

float sgpa;

};

// Function to search and display students by SGPA

void searchBySgpa(student s[], int size, float searchsgpa) {

bool found = false;

cout << "STUDENTS WITH SGPA " << searchsgpa << endl;

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

if (s[i].sgpa == searchsgpa) {

cout << "NAME :: " << s[i].name << endl;

cout << "ROLL NUMBER :: " << s[i].rollno << endl;

cout << "SGPA :: " << s[i].sgpa << endl;

found = true;

}

}

if (!found) {

cout << "NO STUDENTS FOUND WITH SGPA " << searchsgpa << endl;

}

}

// Function to arrange list of students according to roll number using bubble sort

void bubbleSort(student s[], int n) {

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

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

if (s[j].rollno > s[j + 1].rollno) {

swap(s[j], s[j + 1]);

}

}

}

cout << "The list of students sorted by roll no. is as follows :" << endl;

for (int j = 0; j < n; j++) {

cout << "Name :: " << s[j].name << endl;

cout << "Roll no. :: " << s[j].rollno << endl;

cout << "SGPA :: " << s[j].sgpa << endl;

}

cout << "\n";

}

// Function to search a particular name using binary search

void searchByName(student s[], int low, int high, string key) {

while (low <= high) {

int mid = (low + high) / 2;

if (s[mid].name == key) {

cout << "THE POSITION OF " << key << " IN LIST OF STUDENTS IS " << mid << endl;

cout << "Name :: " << s[mid].name << endl;

cout << "Roll no. :: " << s[mid].rollno << endl;

cout << "SGPA :: " << s[mid].sgpa << endl;

return;

} else if (key < s[mid].name) {

high = mid - 1;

} else {

low = mid + 1;

}

}

cout << "STUDENT NOT FOUND" << endl;

}

// Function to arrange list of students alphabetically using insertion sort

void insertionSort(student s[], int n) {

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

student temp = s[i];

int j = i - 1;

while (j >= 0 && s[j].name > temp.name) {

s[j + 1] = s[j];

j--;

}

s[j + 1] = temp;

}

cout << "THE LIST OF STUDENTS ARRANGED ALPHABETICALLY IS :" << endl;

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

cout << "Name :: " << s[i].name << endl;

cout << "Roll no. :: " << s[i].rollno << endl;

cout << "SGPA :: " << s[i].sgpa << endl;

}

}

// Function to partition for quicksort

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) {

i++;

swap(s[i], s[j]);

}

}

swap(s[i + 1], s[high]);

return i + 1;

}

// Function to perform quicksort

void quickSort(student s[], int low, int high) {

if (low < high) {

int pi = partition(s, low, high);

quickSort(s, low, pi - 1);

quickSort(s, pi + 1, high);

}

}

int main() {

int n;

cout << "Enter the number of entries you want to add: ";

cin >> n;

student \*s = new student[n];

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

cout << "Enter Name, Roll no and SGPA of the Student " << i + 1 << endl;

cout << "Enter Name: ";

cin >> s[i].name;

cout << "Enter Roll No: ";

cin >> s[i].rollno;

cout << "Enter SGPA: ";

cin >> s[i].sgpa;

cout << endl;

}

// Display the data

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

cout << "NAME, ROLL NUMBER, SGPA OF STUDENT " << i + 1 << endl;

cout << "NAME :: " << s[i].name << endl;

cout << "ROLL NUMBER :: " << s[i].rollno << endl;

cout << "SGPA :: " << s[i].sgpa << endl;

cout << endl;

}

// Function call to sort by roll number using bubble sort

bubbleSort(s, n);

// Search and display students by SGPA

float searchsgpa;

cout << "ENTER SGPA TO SEARCH: ";

cin >> searchsgpa;

searchBySgpa(s, n, searchsgpa);

// Function call to arrange list of students alphabetically using insertion sort

insertionSort(s, n);

// Function call to search a particular name using binary search

string key;

cout << "ENTER THE NAME OF STUDENT TO SEARCH: ";

cin >> key;

searchByName(s, 0, n - 1, key);

// Function to find 3 toppers using quick sort

quickSort(s, 0, n - 1);

cout << "TOP 3 TOPPERS:" << endl;

for (int i = 0; i < 3 && i < n; i++) {

cout << "NAME :: " << s[i].name << endl;

cout << "ROLL NO :: " << s[i].rollno << endl;

cout << "SGPA :: " << s[i].sgpa << endl;

}

delete[] s;

return 0;

}