#include <iostream>

using namespace std;

struct Student

{

int Roll\_No;

string Name;

float SGPA;

};

int bubblesort(Student s[],int n);

int insertionsort(Student s[],int n);

int quicksort(Student s[], int low, int high);

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

int linearsearch(Student s[], int n);

int binarysearch(Student s[],int n);

int main()

{

cout<<"enter number of entries to accept :";

int n;

cin>>n;

Student s[n];

//input

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

{

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

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

cout<<"Enter Name of Student: ";

cin>>s[i].Name;

cout<<"Enter SGPA of Student: ";

cin>>s[i].SGPA;

cout<<endl;

}

int option;

int func=1;

while(func==1)

{

cout<<"Select function to run"<<endl;

cout<<"Enter 1 for bubble sort (Roll Number Wise)"<<endl<<"Enter 2 for insertion sort (Alphabetical)"<<endl<<"Enter 3 for quick sort (SGPA wise)"<<endl<<"Enter 4 for linear search"<<endl<<"Enter 5 for binary search"<<endl;

cin>>option;

switch (option)

{

case(1):

bubblesort(s,n);

cout<<"Press 1 to continue and 0 to exit"<<endl;

cin>>func;

break;

case(2):

insertionsort(s,n);

cout<<"Press 1 to continue and 0 to exit"<<endl;

cin>>func;

break;

case(3):

quicksort(s, 0, n-1);

//output quicksort

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

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

{

cout<<s[i].Roll\_No<<") "<<s[i].Name<<" "<<s[i].SGPA<<endl;

}

cout<<"Press 1 to continue and 0 to exit"<<endl;

cin>>func;

break;

case(4):

linearsearch(s,n);

cout<<"Press 1 to continue and 0 to exit"<<endl;

cin>>func;

break;

case(5):

quicksort(s, 0, n-1);

binarysearch(s,n);

cout<<"Press 1 to continue and 0 to exit"<<endl;

cin>>func;

break;

default:

break;

}

}

}

//bubble sort

int 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].Roll\_No>s[j+1].Roll\_No)

{

Student temp=s[j];

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

s[j+1]=temp;

}

}

}

//output bubble

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

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

{

cout<<s[i].Roll\_No<<") "<<s[i].Name<<" "<<s[i].SGPA<<endl;

}

}

//insertion sort

int insertionsort(Student s[], int n)

{

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

{

Student tosort = s[i];

int j = i - 1;

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

{

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

j = j - 1;

}

s[j + 1] = tosort;

}

//output insertion

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

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

{

cout<<s[i].Roll\_No<<") "<<s[i].Name<<" "<<s[i].SGPA<<endl;

}

}

//quick sort

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

{

Student pivot = s[high];

int i = low - 1;

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

{

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

{

i++;

Student temp1 = s[i];

s[i] = s[j];

s[j] = temp1;

}

}

Student temp2 = s[i + 1];

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

s[high] = temp2;

return i + 1;

}

int quicksort(Student s[], int low, int high)

{

if (low < high)

{

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

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

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

}

}

//linear search

int linearsearch(Student s[], int n)

{

cout<<endl<<"enter SGPA to search (Linear)"<<endl;

float searchl;

int num=0;

cin>>searchl;

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

{

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

{

cout<<endl<<"Student "<<"Roll No: "<<s[i].Roll\_No<<") "<<s[i].Name<<" "<<s[i].SGPA<<endl;

num++;

}

}

if(num==0)

{

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

}

}

int binarysearch(Student s[], int n)

{

cout<<endl<<"enter SGPA to search (Binary)"<<endl;

float searchb;

int numb=0;

int high=n;

int low=0;

int mid= (high+low)/2;

cin>>searchb;

int found=0;

while(low<=high)

{

int mid= (high+low)/2;

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

{

cout<<endl<<"Student "<<"Roll No: "<<s[mid].Roll\_No<<") "<<s[mid].Name<<" "<<s[mid].SGPA<<endl;

found++;

break;

}

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

{

low=mid+1;

}

else

high=mid-1;

}

if(found==0)

{

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

}