//Sorting and Searching

#include<iostream>

#include<string.h>

using namespace std;

struct student

{

int rn;

char name[50];

float sgpa;

};

void displayinfo(student s[50], int n);

void bubblesort(student s[50], int n); //function declaration

void insertion(student s[50], int n);

void quicksort(student s[50], int first, int last);

void displayinfoReverse(student s[50], int n);

void linearsearch(student s[50], int n, float key);

int binarysearch(student s[50], int low, int high, char keyname[20]);

int main()

{

student s[50];

int i, n, x;

float key;

char keyname[20];

cout<<"How many students data to be entered?\n";

cin>>n;

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

{

cout<<"Enter roll no\n";

cin>>s[i].rn;

cout<<"Enter Name of student\n";

cin>>s[i].name;

cout<<"Enter sgpa\n";

cin>>s[i].sgpa;

}

displayinfo(s, n); //function call

bubblesort(s,n); //function call

quicksort(s, 0, n-1); //function call

displayinfoReverse(s, n); //function call

insertion(s,n); //function call

cout<<"Enter SGPA marks to be searched\n";

cin>>key;

linearsearch(s, n, key);

cout<<"Enter name of the student to be searched\n";

cin>>keyname;

x=binarysearch(s, 0, n-1, keyname);

if(x !=-1)

{

cout<<"student name found at position=\n"<<x;

cout<<"Roll No: "<<s[x].rn<<"\tName :"<<s[x].name<<"\tSGPA:"<<s[x].sgpa;

}

else

cout<<"Student record not found";

return 0;

}

void displayinfo(student s[50], int n)

{

int i;

cout<<"Display student information\n";

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

{

cout<<s[i].rn<<"\t"<<s[i].name<<"\t"<<s[i].sgpa<<"\n";

}

}

void displayinfoReverse(student s[50], int n)

{

int i;

cout<<"Display student information\n";

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

{

cout<<s[i].rn<<"\t"<<s[i].name<<"\t"<<s[i].sgpa<<"\n";

}

}

void bubblesort(student s[50], int n)

{

int i, pass, temp;

char temp1[50];

float temp2;

cout<<"Sort student data as per their roll no\n";

for(pass=1; pass<=n-1; pass++)

{

for(i=0; i<n-pass; i++)

{

if(s[i].rn>s[i+1].rn)

{

temp=s[i].rn;

s[i].rn=s[i+1].rn;

s[i+1].rn=temp;

strcpy(temp1,s[i].name);

strcpy(s[i].name, s[i+1].name);

strcpy(s[i+1].name, temp1);

temp2=s[i].sgpa;

s[i].sgpa=s[i+1].sgpa;

s[i+1].sgpa=temp2;

}

}

}

displayinfo(s,n);

}

void insertion(student s[50], int n)

{

int i, j;

char temp[50];

int temp1;

float temp2;

cout<<"Sorting student information alphabetically\n";

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

{

strcpy(temp,s[i].name);

temp1=s[i].rn;

temp2=s[i].sgpa;

for(j=i-1; j>=0 && (strcmp(temp, s[j].name)<0); j--)

{

strcpy(s[j+1].name, s[j].name);

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

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

}

strcpy(s[j+1].name,temp);

s[j+1].rn=temp1;

s[j+1].sgpa=temp2;

}

displayinfo(s,n);

}

void quicksort(student s[50], int first, int last)

{

int i, j, pivot;

float temp;

int temp1;

char temp2[20];

if(first<last)

{ //pivot

i=first; // 1 2 3 4 5 6

j=last; // 9.2 8.4 8.1 9.5 9.0 9.3

pivot=first; // i j

while(i<j) // 9.2 8.4 8.1 9.0 9.5 9.3

{ // j i

while(s[i].sgpa<=s[pivot].sgpa && i<last)

i++;

while(s[j].sgpa > s[pivot].sgpa)

j--;

if(i<j)

{

temp=s[i].sgpa;

s[i].sgpa=s[j].sgpa;

s[j].sgpa=temp;

temp1=s[i].rn;

s[i].rn=s[j].rn;

s[j].rn=temp1;

strcpy(temp2,s[i].name);

strcpy(s[i].name,s[j].name);

strcpy(s[j].name,temp2);

} // j

} // 9.0 8.4 8.1 9.2 9.5 9.3

temp=s[pivot].sgpa;

s[pivot].sgpa=s[j].sgpa;

s[j].sgpa=temp;

temp1=s[pivot].rn;

s[pivot].rn=s[j].rn;

s[j].rn=temp1;

strcpy(temp2,s[pivot].name);

strcpy(s[pivot].name,s[j].name);

strcpy(s[j].name,temp2);

quicksort(s,first, j-1); //recursive function call left part

quicksort(s, j+1, last); // recursive call for right side

}

}

void linearsearch(student s[50], int n, float key)

{

int i,flag=0;;

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

{

if(key==s[i].sgpa)

{

cout<<"Student got sgpa="<<key<<"is"<<s[i].rn<<"\t"<<s[i].name<<"\n";

flag=1;

}

}

if(flag==0)

cout<<"Student record not found";

}

int binarysearch(student s[50], int low, int high, char keyname[20])

{

int mid;

if(low<=high)

{

mid=(low+high)/2;

if(strcmp(keyname,s[mid].name)==0)

return mid;

else

if(strcmp(keyname,s[mid].name)<0)

return binarysearch(s, low, mid-1, keyname);

else

return binarysearch(s, mid+1, high, keyname);

}

else

return -1;

}