**Practical No.1**

**Searching & Sorting**

**Problem Statement:-**

Consider a student database of SEIT class (at least 15 records). Database contains different fields of every student like Roll No, Name and SGPA.(array of structure)

a) Design a roll call list, arrange list of students according to roll numbers in ascending order (Use Bubble Sort)

b) Arrange list of students alphabetically. (Use Insertion sort)

c) Arrange list of students to find out first ten toppers from a class. (Use Quick sort)

d) Search students according to SGPA. If more than one student having same SGPA, then print list of all students having same SGPA.

e) Search a particular student according to name using binary search without recursion

student records having the presence of search key should be displayed)

(Note: Implement either Bubble sort or Inserti Sort.)

**Program Code:-**

#include<iostream>

#include<string.h>

using namespace std;

const int size = 15;

struct student {

int rno;

char name[20];

float SGPA;

};

void accept(struct student list[size]);

void display(struct student list[size]);

void displayTop(struct student list[size]);

void bubbleSort(struct student list[size]);

void insertSort(struct student list[size]);

void quickSort(struct student list[size], int, int);

void search(struct student list[size]);

void binarysearch(struct student list[size]);

int main() {

int ch;

struct student data[size];

accept(data);

do {

cout << "\n1: Bubble Sort";

cout << "\n2: Insertion Sort";

cout << "\n3: Quick Sort";

cout << "\n4: Search";

cout << "\n5: Binary Search";

cout << "\n6: Exit";

cout << "\nSelect your choice: ";

cin >> ch;

switch (ch) {

case 1:

bubbleSort(data);

display(data);

break;

case 2:

insertSort(data);

display(data);

break;

case 3:

quickSort(data, 0, size - 1);

displayTop(data);

break;

case 4:

search(data);

break;

case 5:

binarysearch(data);

break;

case 6:

cout << "Exiting...\n";

break;

default:

cout << "Invalid choice. Please try again.\n";

}

} while (ch != 6); // Exit on choice 6

return 0;

}

void accept(struct student list[size]) {

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

cout << "Enter roll no., name, and SGPA: ";

cin >> list[i].rno >> list[i].name >> list[i].SGPA;

}

}

void display(struct student list[size]) {

cout << "\n\nRoll No.\tName\tSGPA\n";

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

cout << list[i].rno << "\t\t" << list[i].name << "\t" << list[i].SGPA << "\n";

}

}

void displayTop(struct student list[size]) {

cout << "\n\nRoll No.\tName\tSGPA\n";

for (int i = 0; i < 10 && i < size; i++) { // Ensure top 10 or less if fewer students

cout << list[i].rno << "\t\t" << list[i].name << "\t" << list[i].SGPA << "\n";

}

}

void bubbleSort(struct student list[size]) {

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

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

if (list[j].rno > list[j + 1].rno) {

student temp = list[j];

list[j] = list[j + 1];

list[j + 1] = temp;

}

}

}

}

void insertSort(struct student list[size]) {

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

student temp = list[i];

int j = i - 1;

while (j >= 0 && strcmp(list[j].name, temp.name) > 0) {

list[j + 1] = list[j];

j--;

}

list[j + 1] = temp;

}

}

void quickSort(struct student list[size], int first, int last) {

if (first < last) {

int pivot = first;

int i = first, j = last;

student temp;

while (i < j) {

while (list[i].SGPA >= list[pivot].SGPA && i < last)

i++;

while (list[j].SGPA < list[pivot].SGPA)

j--;

if (i < j) {

temp = list[i];

list[i] = list[j];

list[j] = temp;

}

}

temp = list[pivot];

list[pivot] = list[j];

list[j] = temp;

quickSort(list, first, j - 1);

quickSort(list, j + 1, last);

}

}

void search(struct student list[size]) {

float SGPA;

cout << "Enter SGPA to search: ";

cin >> SGPA;

cout << "\n\nRoll No.\tName\tSGPA\n";

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

if (list[i].SGPA == SGPA) {

cout << list[i].rno << "\t\t" << list[i].name << "\t" << list[i].SGPA << "\n";

}

}

}

void binarysearch(struct student list[size])

{

int k,lower,upper,mid;

char search[80];

cout<<"\n Enter name of student you want to search:-";

cin>>search;

lower=0;

upper=size-1;

mid=(lower+upper)/2;

while(lower<=upper)

{

if(strcmp(list[mid].name,search)<0)

lower=mid+1;

else if(strcmp(list[mid].name,search)==0)

{

cout<<"\n"<<list[mid].rno<<"\t"<<list[mid].name<<"\t"<<list[mid].SGPA;

break;

}

else

upper=mid-1;

mid=(lower+upper)/2;

}

if(lower>upper)

cout<<search<<"Not found in the list";

}