DSA-ASS-10 > C 10_1_student_record.c > ...

```c
/* You are given a text file, named “students.txt” that contains students’ records.
Each Line contains information of a single student in the form of <Student Name, Roll No, Department>.

A. Read the records from the file into an array of structures.

B. Three Options will turn up: (1) Bubble Sort, (2) Binary Search, and (3) Quit.
In the following we describe what your C/C++ program will do on Selecting the options.

(1) Bubble Sort: Sorts the records based on Student Name.
If more than One students has the same name, then sort them on their roll no.

(2) Binary Search: Given a student name, the function will return all the Student records
<Student Name, Roll No, Department> having the Student name.

(3) Quit: Exit the program. */

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define MAXNAME 50
#define MAXDEPT 100
#define MAXNO 100

typedef struct{
    char name[MAXNAME];
    int roll;
    char dept[MAXDEPT];
}Student;

Student students[MAXNO];
int c = 0;
int issorted = 0;

void readfile(){
    FILE *f = fopen("students.txt","r");
    if(!f){
        printf("ERROR : Failed to open File.\n");
        return;
    }
    while(fscanf(f, "%s %d %s", &students[c].name, &students[c].roll, &students[c].dept) == 3){
        c++;
    }
    fclose(f);
}
```

DSA-ASS-10 > C 10_1_student_record.c > ...

```c
void bubblesortname(){
    for(int i = 0; i < c-1; i++){
        for(int j = 0; j < c-i-1; j++){
            if((strcmp(students[j].name,students[j+1].name) > 0) || (strcmp(students[j].name,students[j+1].name) == 0 && students[j].roll > students[j+1].roll)){
                Student temp = students[j];
                students[j] = students[j+1];
                students[j+1] = temp;
            }
        }
    }
    printf("Records Sorted Successfully.\n");
    issorted = 1;
}

void binarysearch(char *x){
    if(!issorted){
        printf("Sort the records first.\n");
        return;
    }
    int l = 0, r = c-1;
    int flag = 0;
    while(l <= r){
        int m = l + (r-l)/2;
        int cmp = strcmp(students[m].name,x);
        if(cmp == 0){
            printf("%s  %d  %s\n", students[m].name, students[m].roll, students[m].dept);
            flag = 1;
        }
        else if(cmp < 0){
            l = m + 1;
        }
        else if(cmp > 0){
            r = m - 1;
        }
    }
    if(!flag){
        printf("ERROR : Given name is not in the records.\n");
    }
}

int main(){
```

```c
int main(){
    readfile();
    if(c == 0){
        printf("No Record Found.\n");
        return 0;
    }
    while(1){
        int ch;
        printf("Enter your choice :--\n1 - Bubble-Sort the records based on Name\n2 - Search for students based on Name\n3 - Exit\nChoice : ");
        scanf("%d",&ch);
        switch(ch){
            case 1: bubblesortname();
                    break;
            case 2:
            {
                char x[MAXNAME];
                printf("Enter Name to search records : ");
                scanf("%s",&x);
                binarysearch(x);
                break;
            }
            case 3: printf("Program Exited.\n");
                    exit(0);
            default: printf("INVALID CHOICE - TRY AGAIN.\n");
        }
    }
    return 0;
}
```

DSA-ASS-10 > students.txt

```
1  Ritweek 1 AI
2  Neel 2 AI
3  Shuvrangshu 3 AI
4  Nischay 4 CSE
5  Neel 5 CSE
6  Zaid 6 Mechanical
7  Ritweek 7 AI
8  Neel 8 Biology
9  Hitesh 9 ECE
10 Ved 10 MNC
```

```
Enter your choice :--
1 - Bubble-Sort the records based on Name
2 - Search for students based on Name
3 - Exit
Choice : 1
Records Sorted Successfully.

Enter your choice :--
1 - Bubble-Sort the records based on Name
2 - Search for students based on Name
3 - Exit
Choice : 2
Enter Name to search records : Shuvrangshu
Shuvrangshu  3  AI

Enter your choice :--
1 - Bubble-Sort the records based on Name
2 - Search for students based on Name
3 - Exit
Choice : 3
Program Exited.
PS C:\Users\shuvr\OneDrive\Documents\CODING\College C codes\SEM-2>
```

DSA-ASS-10 > C 10_2_inversion.c > inversionupdate(int *, int, int, int)

```c
/* Let A[n] be an array of n distinct integers.
If i < j and A[i] > A[j], then the pair (i, j) Is called an inversion of A.
Write a C/C++ program that determines the number of Inversions in any permutation on n elements in O(n lg n) worst-case time.
(Hint: Modify merge sort)
Example: A = {4, 1, 3, 2} output is 4 */

#include <stdio.h>
#include <stdlib.h>

int c = 0;

void inversionupdate(int *A, int l, int m, int r)
{
    int i = l, j = m + 1, k = 0;
    int B[r - l + 1];
    while (i <= m && j <= r)
    {
        if (A[i] <= A[j])
        {
            B[k++] = A[i++];
        }
        else
        {
            B[k++] = A[j++];
            c += (m - i + 1); // All remaining elements in left subarray form an inversion
        }
    }
    while (i <= m)
    {
        B[k++] = A[i++];
    }
    while (j <= r)
    {
        B[k++] = A[j++];
    }
    for (int _ = 0; _ < k; _++)
    {
        A[l + _] = B[_];
    }
}

```

```c
void inversionC(int *A, int l, int r)
{
    if (l < r)
    {
        int m = (l + r) / 2;
        inversionC(A, l, m);
        inversionC(A, m + 1, r);
        inversionupdate(A, l, m, r);
    }
}

int main()
{
    int ch;
    while (1)
    {
        printf("\nEnter your Choice :--\n1 - Check the number of inversion pairs in an array\n2 - Quit\nChoice : ");
        scanf("%d", &ch);
        switch (ch)
        {
        case 1:
        {
            int n;
            printf("\nEnter size of array : ");
            scanf("%d", &n);
            int A[n];
            printf("Enter %d numbers : ", n);
            for (int i = 0; i < n; i++)
            {
                scanf("%d", &A[i]);
            }
            inversionC(A, 0, n - 1);
            printf("No. of inversion pairs : %d\n", c);
            printf("Sorted Array : ");
            for (int i = 0; i < n; i++)
            {
                printf("%d ", A[i]);
            }
            printf("\n");
            c = 0;
            break;
        }
        case 2:
            exit(0);
        default:
            printf("INVALID INPUT - TRY AGAIN.\n");
        }
    }
    return 0;
}
```

```
Enter your Choice :--
1 - Check the number of inversion pairs in an array
2 - Quit
Choice : 1

Enter size of array : 4
Enter 4 numbers : 4 1 3 2
No. of inversion pairs : 4
Sorted Array : 1 2 3 4

Enter your Choice :--
1 - Check the number of inversion pairs in an array
2 - Quit
Choice : 1

Enter size of array : 10
Enter 10 numbers : 1 4 2 6 3 7 4 10 9 8
No. of inversion pairs : 8
Sorted Array : 1 2 3 4 4 6 7 8 9 10

Enter your Choice :--
1 - Check the number of inversion pairs in an array
2 - Quit
Choice : 2
PS C:\Users\shuvr\OneDrive\Documents\CODING\College C codes\SEM-2>
```