



**Symbiosis Institute of Technology, Pune**

**Department of Computer Science and Engineering**

**Academic Year 2025-26**

**Design and Analysis of Algorithms– Lab**

**Batch 2023-27 - Sem V**

| Lab Assignment No:- 1                          |  |
|--|--|
| <b>Name of Student</b>                         | Deepti Pal   |
| <b>PRN No.</b>                                 | 23070122081  |
| <b>Batch</b>                                   | TY CSE (2023-27)   |
| <b>Class</b>                                   | A3   |
| <b>Academic Year &amp; Semester</b>            | 2025-26, Third Year, 5 <sup>th</sup> Semester  |
| <b>Date of Submission</b>                      | 3 August 2025  |
| <b>Title of Assignment:</b>                    | <p>WAP to search an element using linear search, binary search (recursive) and analyze their complexity.</p> <p><b>Problem Given:</b></p> <ul style="list-style-type: none"> <li>Create a student records file, with 5 columns: PRN, student_name, class, mobile_number, and marks. Let the PRN's be in ascending order. Write an interactive program to search student record given any of the data. Use linear search or binary search as applicable. The student record should have at least 25 records.</li> </ul> |
| <b>Theory: (Handwritten)</b>                   | 1. Compare Linear search and binary search algorithms.   |
| <b>Source code (Implementation Screenshot)</b> | <pre>#include&lt;stdio.h&gt; #include&lt;string.h&gt;  typedef struct {     int prn;     char name[50];     char class[10];     int mobile_no;     int marks;</pre>  |

```

} Student;

//linear search
int linearSearch(Student arr[], int n, char* charTarget, int field) {
    for(int i=0; i<n; i++) {
        if(field == 2 && strcmp(arr[i].name, charTarget) == 0) {
            return i;
        }
        else if(field == 3 && strcmp(arr[i].class, charTarget) == 0) {
            return i;
        }
    }
    return -1;
}

//binary search
int binarySearch(Student arr[], int n, int target, int field) {
    int low = 0;
    int high = n;
    while(low <= high) {
        int mid = (low + high)/2;
        int midVal = (field == 1)? arr[mid].prn :
            (field == 4)? arr[mid].mobile_no :
            arr[mid].marks;
        if(midVal == target) {
            return mid;
        }
        else if( target < midVal) {
            high = mid - 1;
        }
        else {
            low = mid + 1;
        }
    }
    return -1;
}

int main() {
    FILE *fp = fopen("TestFile_a1.txt", "r");
    char line[256];
    char *cell;
    char charTarget[50];
    int inputType, intTarget, result=-1;
    Student arr[30];

    if(fp == NULL) {
        printf("File Not found!");
        return 1;
    }

    fgets(line, sizeof(line), fp); // to skip the header line

    int c=0;
    while(fgets(line, sizeof(line), fp) && c < 30) {

```

```

        int s_no;
        sscanf(line, "%d %d %s %s %d %d",
                &s_no,
                &arr[c].prn,
                arr[c].name,
                arr[c].class,
                &arr[c].mobile_no,
                &arr[c].marks);
        c++;
    }

    printf("Enter your choice of input to search: \n");
    printf("1)Student PRN\n");
    printf("2)Student Name\n");
    printf("3)Student class\n");
    printf("4)Student Ph.no\n");
    printf("5)Student marks\n");
    scanf("%d", &inputType);

    switch(inputType) {
    case 1: {
        printf("Enter target prn: ");
        scanf("%d", &intTarget);
        result = binarySearch(arr, c, intTarget, 1);
        break;
    }
    case 2: {
        printf("Enter target name: ");
        scanf("%s", charTarget);
        result = linearSearch(arr, c, charTarget, 2);
        break;
    }
    case 3: {
        printf("Enter target class: ");
        scanf("%s", charTarget);
        result = linearSearch(arr, c, charTarget, 3);
        break;
    }
    case 4: {
        printf("Enter target Ph.no.: ");
        scanf("%d", &intTarget);
        result = binarySearch(arr, c, intTarget, 4);
        break;
    }
    case 5: {
        printf("Enter target marks: ");
        scanf("%d", &intTarget);
        result = binarySearch(arr, c, intTarget, 5);
        break;
    }
    }

    if(result<0) {
        printf("Details of target student not found !!");
    }

```

```

    } else {
        printf("Details of the student are in row of s.no = %d",
result+1);
    }

    fclose(fp);
    return 0;
}

```

Input file:

| main.c |      | TestFile_a1.txt : |              |       |           |       |
|--------|------|-------------------|--------------|-------|-----------|-------|
|        | s.no | prn               | student name | class | mobile no | marks |
| 1      |      |                   |              |       |           |       |
| 2      | 1    | 31                | akshay       | III   | 92681     | 59    |
| 3      | 2    | 32                | krishna      | IV    | 81723     | 52    |
| 4      | 3    | 33                | ridhi        | V     | 81919     | 88    |
| 5      | 4    | 34                | ayush        | III   | 62726     | 96    |
| 6      | 5    | 35                | priya        | VI    | 45258     | 48    |
| 7      | 6    | 36                | radhika      | III   | 81735     | 26    |
| 8      | 7    | 37                | priyanka     | IV    | 89121     | 58    |
| 9      | 8    | 38                | dipesh       | V     | 12340     | 52    |
| 10     | 9    | 39                | aarushi      | V     | 89119     | 37    |
| 11     | 10   | 40                | nidhi        | VI    | 99118     | 84    |
| 12     | 11   | 41                | shubham      | III   | 93214     | 75    |
| 13     | 12   | 42                | mehak        | IV    | 78632     | 64    |
| 14     | 13   | 43                | vivek        | V     | 81234     | 45    |
| 15     | 14   | 44                | arjun        | VI    | 74325     | 89    |
| 16     | 15   | 45                | sneha        | III   | 89674     | 67    |
| 17     | 16   | 46                | ananya       | IV    | 90123     | 79    |
| 18     | 17   | 47                | dev          | V     | 87456     | 33    |
| 19     | 18   | 48                | tanvi        | VI    | 82345     | 92    |
| 20     | 19   | 49                | manav        | III   | 85421     | 60    |
| 21     | 20   | 50                | ishita       | IV    | 87654     | 50    |

## Output Screenshots

```
main.c | TestFile_a1.txt |
1  #include<stdio.h>
2  #include<string.h>
3
4
5  typedef struct {
6      int prn;
7      char name[50];
8      char class[10];
9      int mobile_no;
10     int marks;
11 } Student;
12
13 //Linear search
14 int linearSearch(Student arr[], int n, char* charTarget, int field) {
```

input

Enter your choice of input to search:  
1) Student PRN  
2) Student Name  
3) Student class  
4) Student Ph.no  
5) Student marks  
1  
Enter target prn: 45  
Details of the student are in row of s.no = 15  
...Program finished with exit code 0  
Press ENTER to exit console.

```
main.c | TestFile_a1.txt |
1  #include<stdio.h>
2  #include<string.h>
3
4
5  typedef struct {
6      int prn;
7      char name[50];
8      char class[10];
9      int mobile_no;
10     int marks;
11 } Student;
12
13 //Linear search
14 int linearSearch(Student arr[], int n, char* charTarget, int field) {
```

input

Enter your choice of input to search:  
1) Student PRN  
2) Student Name  
3) Student class  
4) Student Ph.no  
5) Student marks  
2  
Enter target name: shubham  
Details of the student are in row of s.no = 11  
...Program finished with exit code 0

to  
arch  
have

```
main.c TestFile_a1.txt
1 #include<stdio.h>
2 #include<string.h>
3
4
5 typedef struct {
6     int prn;
7     char name[50];
8     char class[10];
9     int mobile_no;
10    int marks;
11 } Student;
12
13 //linear search
14 int linearSearch(Student arr[], int n, char* charTarget
```

input

Enter your choice of input to search:

1) Student PRN  
2) Student Name  
3) Student class  
4) Student Ph.no  
5) Student marks

1

Enter target prn: 12

Details of target student not found !!

...Program finished with exit code 0  
Press ENTER to exit console.

## Problems Solved from Hacker Rank (Minimum 4)

1. <https://www.hackerrank.com/challenges/icecream-parlor/problem?isFullScreen=true>

### Solution Screenshot:

```
HackerRank Prepare > Algorithms > Search > Ice Cream Parlor
```

all of the money they have.

Example.  $m = 6$  cost = [1, 3, 4, 5, 6]

The two flavors that cost 1 and 5 meet the criteria. Using 1-based indexing, they are at indices 1 and 4.

**Function Description**

Complete the icecreamParlor function in the editor below.

icecreamParlor has the following parameter(s):

- int m: the amount of money they have to spend
- int cost[]: the cost of each flavor of ice cream

**Returns**

- int[]: the indices of the prices of the two flavors they buy, sorted ascending

**Input Format**

The first line contains an integer,  $t$ , the number of trips to the ice cream parlor. The next  $t$  sets of lines each describe a visit.

Each trip is described as follows:

1. The integer  $m$ , the amount of money they have pooled.
2. The integer  $n$ , the number of flavors offered at the time.
3.  $n$  space-separated integers denoting the cost of each flavor: cost[0], cost[1], ..., cost[n].

```
18 vector<int> icecreamParlor(int m, vector<int> arr) {
19     vector<pair<int,int>> v; // {value, original_index}
20     for (int i = 0; i < arr.size(); i++)
21         v.push_back({arr[i], i+1}); // store 1-based index
22
23     sort(v.begin(), v.end()); // sort by cost automatically
24
25     int low = 0, high = v.size()-1;
26     while (low < high) {
27         int sum = v[low].first + v[high].first;
28         if (sum == m) {
29             // return original indices in ascending order
30             return {min(v[low].second, v[high].second),
31                     max(v[low].second, v[high].second)};
32         }
33         else if (sum > m) {
34             high--;
35         }
36         else {
37             low++;
38         }
39     }
40     return {-1};
41 }
```

Line: 39 Col: 17

Upload Code as File Test against custom input Run Code Submit Code

## Congratulations

You solved this challenge. Would you like to challenge your friends? [f](#) [t](#) [in](#)

[Next Challenge](#)

✓ Test case 0

✓ Test case 1 [🔒](#)

✓ Test case 2 [🔒](#)

✓ Test case 3

Compiler Message

Success

Input (stdin)

[Download](#)

```
1 2
2 4
3 5
4 1 4 5 3 2
5 4
6 4
7 2 2 4 3
```

Expected Output

[Download](#)

2. <https://www.hackerrank.com/challenges/missing-numbers/problem?isFullScreen=true>

### Solution Screenshot:

**Problem Description:**

Given two arrays of integers, find which elements in the second array are missing from the first array.

**Example**

```
arr = [7, 2, 5, 3, 5, 3]
brr = [7, 2, 5, 4, 6, 5, 3]
```

The `brr` array is the original list. The numbers missing are [4, 6].

**Notes**

- If a number occurs multiple times in the lists, you must ensure that the frequency of that number in both lists is the same. If that is not the case, then it is also a missing number.
- Return the missing numbers sorted ascending.
- Only include a missing number once, even if it is missing multiple times.
- The difference between the maximum and minimum numbers in the original list is less than or equal to 100.

**Function Description**

Complete the `missingNumbers` function in the editor below. It should return a sorted array of missing numbers.

`missingNumbers` has the following parameter(s):

- `int arr[]`: the array with missing numbers
- `int brr[]`: the original array of numbers

**Returns**

A sorted array of missing numbers.

**C++ Code:**

```
14  * 1. INTEGER_ARRAY arr
15  * 2. INTEGER_ARRAY brr
16  */
17
18  vector<int> missingNumbers(vector<int> arr, vector<int> brr) {
19      unordered_map<int, int> freqArr, freqBrr;
20
21      // Count frequencies
22      for (int x : arr) freqArr[x]++;
23      for (int x : brr) freqBrr[x]++;
24
25      vector<int> result;
26      for (auto &p : freqBrr) {
27          int num = p.first;
28          if (freqBrr[num] > freqArr[num]) {
29              result.push_back(num);
30          }
31      }
32
33      sort(result.begin(), result.end()); // output should be sorted
34      return result;
35  }
```





connected to be the same color in the region.

Given an  $n \times m$  matrix, find and print the number of cells in the largest region in the matrix. Note that there may be more than one region in the matrix.

For example, there are two regions in the following  $2 \times 3$  matrix. The larger region at the top left contains 3 cells. The smaller one at the bottom right contains 1.

```
110
100
001
```

**Function Description**

Complete the `connectedCell` function in the editor below.

```
45         int regionSize = dfs(matrix, i, j, visited);
46         maxRegion = max(maxRegion, regionSize);
47     }
48 }
49
50 return maxRegion;
51
52
53 int main()
54 {
```

Line: 42 Col: 34

[Upload Code as File](#) ☐ Test against custom input [Run Code](#) [Submit Code](#)

## Congratulations

You solved this challenge. Would you like to challenge your friends? [f](#) [t](#) [in](#)

[Next Challenge](#)

✔ Test case 0

✔ Test case 1

✔ Test case 2

✔ Test case 3

✔ Test case 4

✔ Test case 5

✔ Test case 6

Compiler Message

Success

Input (stdin) [Download](#)

```
1 5
2 5
3 1 1 0 0 0
4 0 1 1 0 0
5 0 0 1 0 1
6 1 0 0 0 1
7 0 1 0 1 1
```

Expected Output [Download](#)