INDIRA GANDHI DELHI TECHNICAL UNIVERSITY FOR WOMEN

DATA STRUCTURES

(BCS - 201)

DATA STRUCTURES LABARATORY FILE

SUBMITTED BY:

DONE BY: Venkata Shreya Jakkinapalli

ENROLLMENT NO.: 22101012022

COURSE: Bachelors Of Technology

DEPARTMENT: Computer Science And Engineering (CSE -3)

BATCH: 2022-2026

SUBMITTED TO:

Prof. SRN Reddy

Professor

CSE Dept., IGDTUW

TABLE OF CONTENTS

LAB 1

List Of Experiments

Q:1) Linear Search

Q:2) Finding Largest And Smallest Element

Q:3) Merge Two Arrays

Q:1) LINEAR SEARCH

Section 1:

AIM: To perform linear search to search for an element in an array

Section 2:

Software And Hardware Requirements

- Software:
- Hardware:

Section 3:

Algorithm/ Pseudocode

```
1. Start
2. Initialize an integer array 'arr' of size 10.
3. Display "ENTER NUMBER OF ELEMENTS IN ARRAY: "
4. Read the integer 'N' from the user.
5. For 'i' from 0 to N-1:
        a. Display "Enter element: "
        b. Read 'arr[i]' from the user.
6. Display "Enter element to search: "
7. Read the integer 'ele' from the user.
8. Initialize an integer 'ans' to -1. This variable will store the index of the found element.
9. For 'i' from 0 to N-1:
    a. If 'arr[i]' is equal to 'ele':
      - Set 'ans' to 'i'.
      - Break out of the loop.
10. If 'ans' is equal to -1:
    a. Display "Element not found!"
11. Else:
        a. Display "Element found at index: " + 'ans'.
    12. End
```

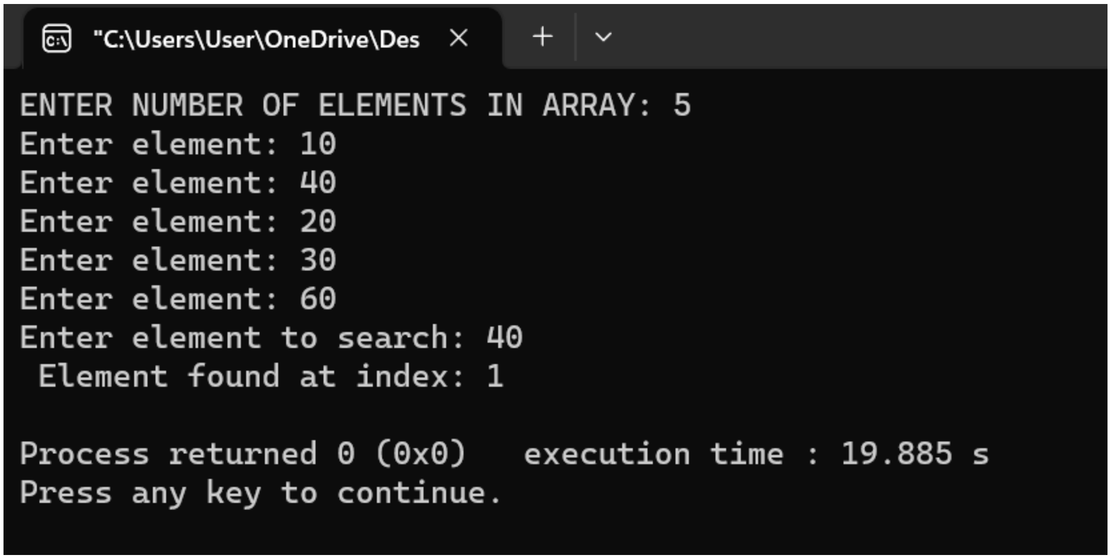
Section 4:

Code, Input And Output

SOURCE CODE:

```
using namespace std;
int search_ele(int arr[], int N, int ele){
    for(int i =0; i <N; i++){
        if(arr[i] == ele ){
            return i;
            break;
        } else{
            continue;
        }
    }
    return -1;
}
int main()
{
    int N;
    int arr[10];
    cout << "ENTER NUMBER OF ELEMENTS IN ARRAY: " ;
    cin >> N;
    for(int i =0; i<N; i++){
        cout << "Enter element: " ;
        cin >> arr[i];
    }
    int ele;
    cout << "Enter element to search: " ;
    cin >> ele;
    int ans = search_ele(arr, N, ele);
    if( ans == -1){
        cout << " Element not found! "<< endl;
    } else{
        cout << " Element found at index: " << ans << endl;
    }
    return 0;
}
```

OUTPUT:



```
ENTER NUMBER OF ELEMENTS IN ARRAY: 5
Enter element: 10
Enter element: 40
Enter element: 20
Enter element: 30
Enter element: 60
Enter element to search: 40
 Element found at index: 1

Process returned 0 (0x0)   execution time : 19.885 s
Press any key to continue.
```

Section 5:

Observations:

- Linear Searching Algorithms: The code implements the Linear Search algorithm, which is a simple searching technique to find the position of a target element within an array.
- It iterates through the array elements one by one until it finds the target element or reaches the end of the array.
- After inputting the array elements, the user is asked to enter the element they want to search for (denoted as 'ele')

- It uses C++ standard input/output (cin/cout) for user interaction.
- The linear search function 'search_ele' could be reused in other programs to perform similar search operations on arrays.

Q:2) Largest And Smallest Element

Section 1:

To find the minimum and maximum elements in an array of integers.

Section 2:

Software And Hardware Requirements

- Software:
- Hardware:

Section 3:

Algorithm/ Pseudocode

```
Function find_maxmin(N, arr)

Initialize variables:

N as an integer (number of elements in the array)

arr as an integer array (input array of integers)

For i from 0 to N - 1:

For j from i + 1 to N - 1:

If arr[i] is greater than arr[j], then:

Swap arr[i] and arr[j] using a temporary variable x.

End of function

Function main



Declare variables:

N as an integer (number of elements in the array)

arr[10] as an integer array (maximum size of 10 elements)

Output "ENTER NUMBER OF ELEMENTS IN ARRAY: "
```

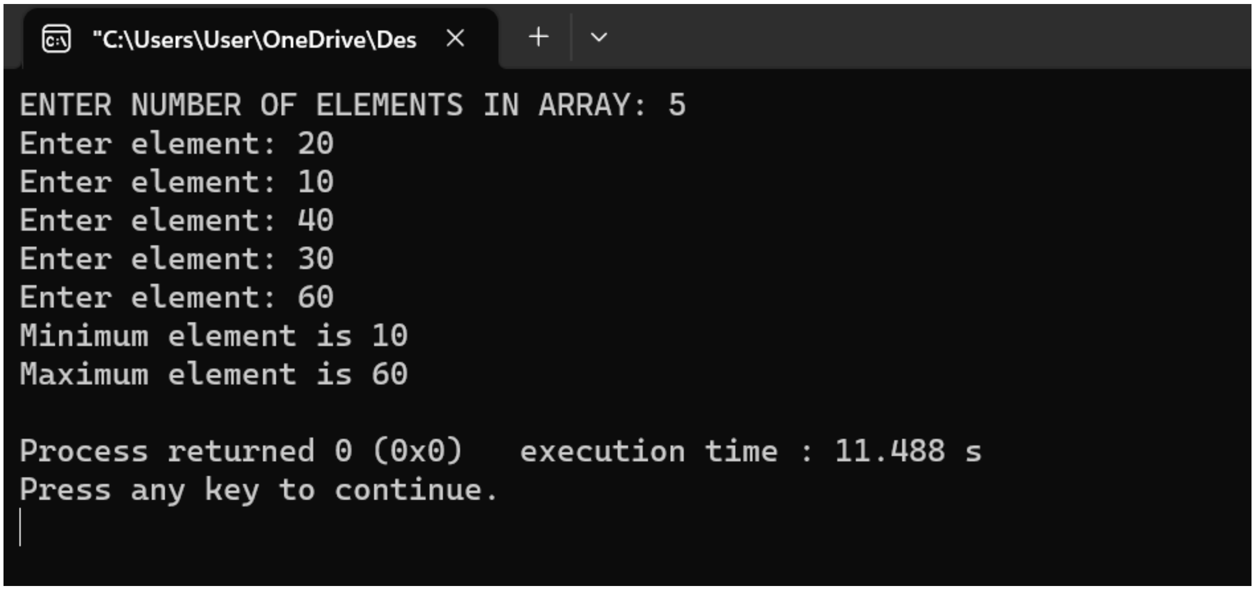
```
Input N (number of elements in the array)
For i from 0 to N - 1:
Output "Enter element: "
Input arr[i] (input each element of the array)
Call find_maxmin(N, arr) to find the minimum and maximum elements in the array.
Output "Minimum element is " followed by the first element of the sorted array (arr[0]).
Output "Maximum element is " followed by the last element of the sorted array (arr[N-1]).
End of function
```

SOURCE CODE:

```
using namespace std;
int find_maxmin(int N, int arr[]){
   for(int i =0; i<N;i++){
      for(int j =i+1;j<N;j++){
         if(arr[i]>arr[j]){
            int x = arr[i];
            arr[i]=arr[j];
            arr[j]=x;
         }
      }
   }
}
int main()
```

```
{
    int N;
    int arr[10];
    cout << "ENTER NUMBER OF ELEMENTS IN ARRAY: " ;
    cin >> N;
    for(int i =0; i<N; i++){
        cout << "Enter element: " ;
        cin >> arr[i];
    }
    int ans = find_maxmin(N,arr);
    cout << "Minimum element is " << arr[0] << endl;
    cout << "Maximum element is " << arr[N-1] << endl;
   return 0;
}
```

OUTPUT:



```
ENTER NUMBER OF ELEMENTS IN ARRAY: 5
Enter element: 20
Enter element: 10
Enter element: 40
Enter element: 30
Enter element: 60
Minimum element is 10
Maximum element is 60

Process returned 0 (0x0)   execution time : 11.488 s
Press any key to continue.
```

Section 5:

Observations:

- To find the minimum and maximum elements, the code uses a simple sorting algorithm. It performs a nested loop with two iterators 'i' and 'j' to compare and swap elements if they are out of order (ascending order).
- After sorting the array in ascending order, the minimum element is the first element in the sorted array (index 0), and the maximum element is the last element (index N-1).

Q:3) Merge Two Sorted Arrays

Section 1:

Aim: To merge 2 arrays containing integers

Section 2:

Software And Hardware Requirements

- Software:
- Hardware:

Section 3:

Algorithm/ Pseudocode

```
Function merge_array(arr1, arr2, N, M, res)
Initialize variables:
i to 0 (for array arr1 traversal)
j to 0 (for array arr2 traversal)
k to 0 (for result array res indexing)
While i is less than N and j is less than M:
If arr1[i] is less than or equal to arr2[j], then:
Assign arr1[i] to res[k]
Increment k by 1
Increment i by 1
Otherwise (if arr2[j] is less than arr1[i]):
Assign arr2[j] to res[k]
Increment k by 1
Increment j by 1
```

```
    While i is less than N:
        Assign arr1[i] to res[k]
        Increment k by 1
        Increment i by 1
    While j is less than M:
        Assign arr2[j] to res[k]
        Increment k by 1
        Increment j by 1
End of function
Function main

    Declare variables:
    N and M as integers
    arr1[10] and arr2[10] as integer arrays
    mergedArr[20] as an integer array (you can adjust the size as needed)
    Output "ENTER NUMBER OF ELEMENTS IN ARRAY 1: "
    Input N
    For i from 0 to N - 1:
        Output "Enter element: "
        Input arr1[i]
    Output "ENTER NUMBER OF ELEMENTS IN ARRAY 2: "
    Input M
    For i from 0 to M - 1:
```

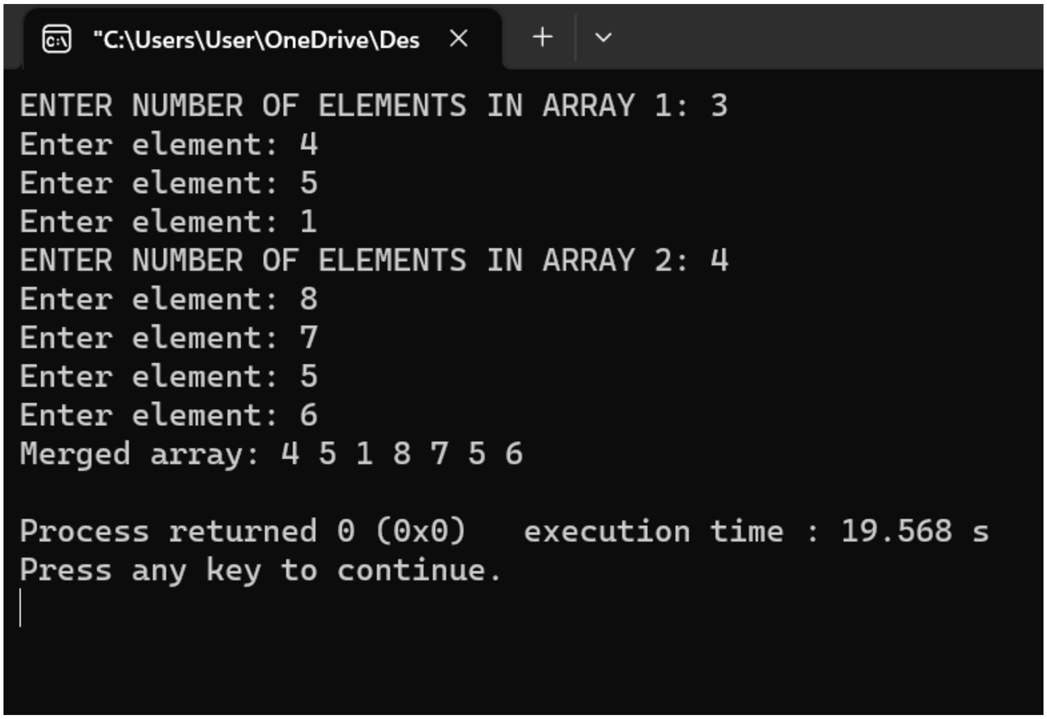
```
Output "Enter element: "
Input arr2[i]
Calculate mergedSize as N + M
Call merge_array(arr1, arr2, N, M, mergedArr)
Output "Merged array: "
For i from 0 to mergedSize - 1:
Output mergedArr[i], " "
Output newline
End of function
```

Section 4:

SOURCE CODE:

```
using namespace std;
void merge_array(int arr1[], int arr2[], int N, int M, int res[]) {
    int i = 0;
    int j = 0;
    int k = 0;
    while (i < N && j < M) {
        if (arr1[i] <= arr2[j]) {
            res[k++] = arr1[i++];
        } else {
            res[k++] = arr2[j++];
        }
    }
    while (i < N) {
        res[k++] = arr1[i++];
    }
    while (j < M) {
        res[k++] = arr2[j++];
    }
}
int main() {
    int N;
    int M;
    int arr1[10];
    int arr2[10];
    cout << "ENTER NUMBER OF ELEMENTS IN ARRAY 1: ";
    cin >> N;
    for (int i = 0; i < N; i++) {
        cout << "Enter element: ";
        cin >> arr1[i];
    }

    cout << "ENTER NUMBER OF ELEMENTS IN ARRAY 2: ";
    cin >> M;
    for (int i = 0; i < M; i++) {
        cout << "Enter element: ";
        cin >> arr2[i];

    }

    int mergedSize = N + M;

    int mergedArr[20]; // You can adjust the size as needed

    merge_array(arr1, arr2, N, M, mergedArr);

    cout << "Merged array: ";

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

        cout << mergedArr[i] << " ";

    }

    cout << "\n";

    return 0;

}
```

OUTPUT:



```
ENTER NUMBER OF ELEMENTS IN ARRAY 1: 3
Enter element: 4
Enter element: 5
Enter element: 1
ENTER NUMBER OF ELEMENTS IN ARRAY 2: 4
Enter element: 8
Enter element: 7
Enter element: 5
Enter element: 6
Merged array: 4 5 1 8 7 5 6

Process returned 0 (0x0)   execution time : 19.568 s
Press any key to continue.
```

Section 5:

Observations:

- The merging logic is implemented using a merge_array function. It iterates through both sorted arrays, comparing elements, and merging them into a third array named mergedArr.
- The code is structured with a merge_array function responsible for the merging logic, and the main function for user input and result display.
- It uses C++ standard input/output (cin/cout) for user interaction.