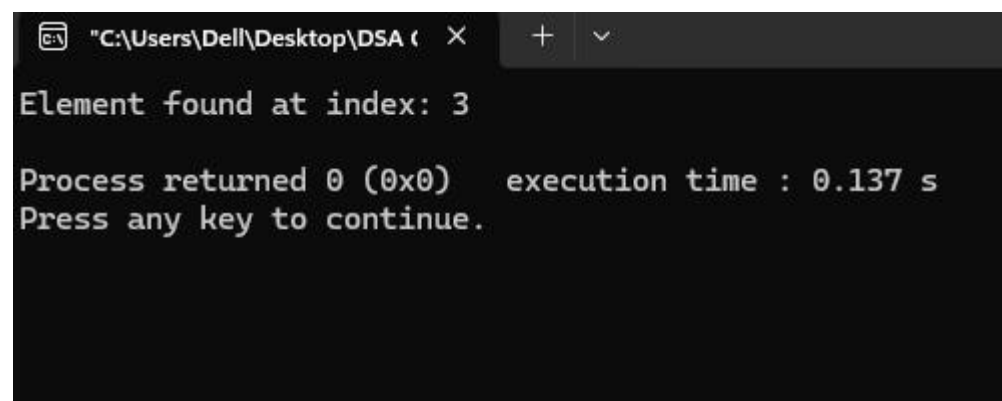


Experiment No: 07

Experiment Name: Searching Algorithm: 1. Linear Search 2. Binary Search

Source Code:**1. Linear Search**

```
#include <stdio.h>
int search(int array[], int size, int value) {
    for (int i = 0; i < size; i++) {
        if (array[i] == value) {
            return i;
        }
    }
    return -1;
}
int main() {
    int array[] = {2, 4, 0, 1, 9};
    int value = 1;
    int size = sizeof(array) / sizeof(array[0]);
    int position = search(array, size, value);
    if (position == -1) {
        printf("Element not found.\n");
    } else {
        printf("Element found at index: %d\n", position);
    }
    return 0;
}
```

Output:A screenshot of a Windows command prompt window. The title bar shows the file path "C:\Users\Dell\Desktop\DSA (". The window content displays the output of the program: "Element found at index: 3". Below this, it shows "Process returned 0 (0x0) execution time : 0.137 s" and "Press any key to continue.".

```
"C:\Users\Dell\Desktop\DSA (  +  v
Element found at index: 3
Process returned 0 (0x0)  execution time : 0.137 s
Press any key to continue.
```

2. Binary Search

```
#include <stdio.h>
int main()
{

    int c, first, last, middle, n, search, array[100];

    printf("Enter number of elements\n");
    scanf("%d", &n);

    printf("Enter %d integers\n", n);
    for (c = 0; c < n; c++)
        scanf("%d", &array[c]);

    printf("Enter value to find\n");
    scanf("%d", &search);

    first = 0;
    last = n - 1;
    middle = (first + last) / 2;
    while (first <= last) {
        if (array[middle] < search)
            first = middle + 1;

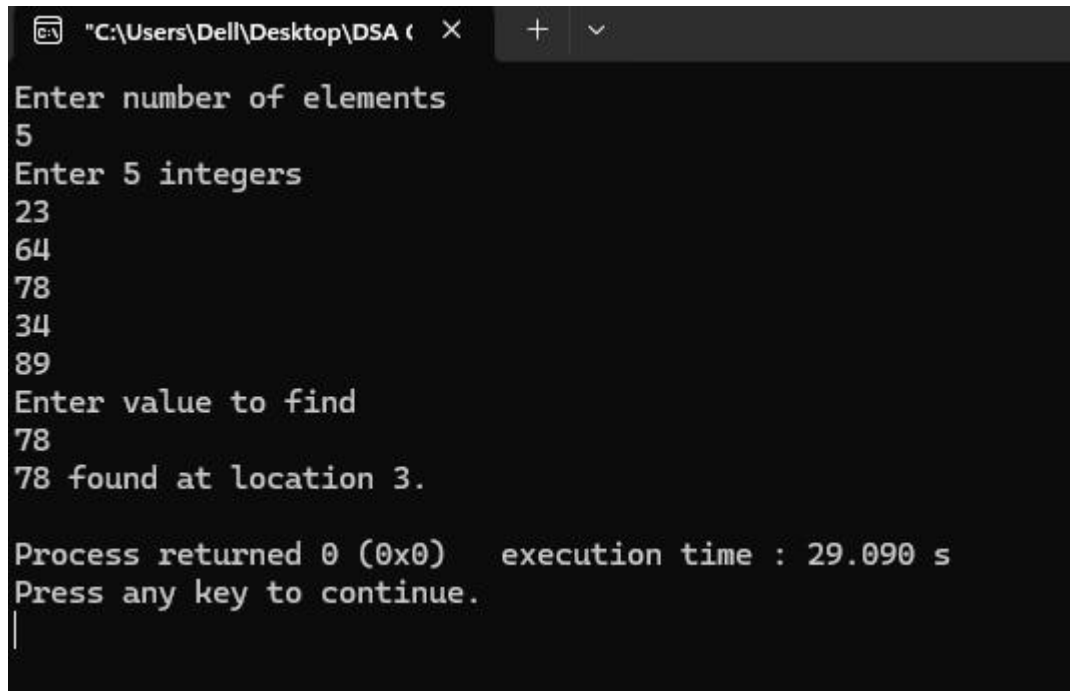
        else if (array[middle] == search) {
            printf("%d found at location %d.\n", search, middle + 1);
            break;
        }

        else
            last = middle - 1;
        middle = (first + last) / 2;
    }

    if (first > last)
        printf("Not found! %d isn't present in the list.\n", search);

    return 0;
}
```

Output:



```
"C:\Users\Dell\Desktop\DSA ( X
Enter number of elements
5
Enter 5 integers
23
64
78
34
89
Enter value to find
78
78 found at location 3.

Process returned 0 (0x0)   execution time : 29.090 s
Press any key to continue.
|
```

Discussion: In this experiment, we applied two common searching techniques: linear search and binary search. The linear search checks each element in the array one by one until it finds the desired value or reaches the end. It is simple and works well for small or unsorted arrays. On the other hand, binary search is more efficient but requires the array to be sorted. It repeatedly divides the search range in half, comparing the middle element with the target. If the value is found, its position is returned; otherwise, the search continues in the relevant half. Both methods were implemented and tested successfully. The program gave the correct output for different inputs, confirming that the search logic worked as expected.