LAB Assignment Submission

for

Data Structures and Algorithms

Course Code: CSE2711

B.Tech CSE-VII/ECOM

Batch 2024

Name: Vedansh Mathur

Enrollment Number: 240686

Problem Statement-

1. Implement Array data-structure (1-dimensional) (write a separate function for each of

the operation):

Populate the array with some data

Insert a particular element at index i

Delete the element at index i

Traverse the array and display the content of the array

Search for a particular element with index/subscript

Print the address of the elements to validate the fact that the elements of an array

are stored in consecutive memory locations.

Check whether your computer stores a 2-dimensional array in row-major or column-

major order.

2. Deβine a 1D array. Populate it with 20 numbers in unsorted order.

i. Search for a particular element in the array by implementing

Linear/Sequential search algorithm.

ii. Compute the running time of the algorithm.

Run the code and display the output.

3. Define a 1D array. Populate it with 10 numbers in sorted order. Search for a particular

element in the array by implementing the Binary search algorithm.

i. Implement Binary search using both iterative and recursive approaches.

ii. Compute the running time of the algorithm.

Run the code and display the output.

Solution -

Q1->

```cpp
#include <iostream>
using namespace std;

const int MAX = 100;   // maximum size of array

// 1. Populate array
void populateArray(int arr[], int &n) {
    cout << "Enter number of elements: ";
    cin >> n;
    cout << "Enter elements: ";
    for (int i = 0; i < n; i++)
        cin >> arr[i];
}

// 2. Insert at index
void insertAtIndex(int arr[], int &n, int index, int value) {
    if (n >= MAX || index < 0 || index > n) {
        cout << "Insertion not possible!\n";
        return;
    }
    for (int i = n; i > index; i--)
        arr[i] = arr[i - 1];
    arr[index] = value;
    n++;
}

// 3. Delete at index
void deleteAtIndex(int arr[], int &n, int index) {
    if (index < 0 || index >= n) {
        cout << "Deletion not possible!\n";
        return;
    }
    for (int i = index; i < n - 1; i++)
        arr[i] = arr[i + 1];
    n--;
}
```

```cpp
// 4. Traverse (display array)
void traverseArray(int arr[], int n) {
    cout << "Array elements: ";
    for (int i = 0; i < n; i++)
        cout << arr[i] << " ";
    cout << endl;
}

// 5. Search element
void searchElement(int arr[], int n, int key) {
    for (int i = 0; i < n; i++) {
        if (arr[i] == key) {
            cout << "Element " << key << " found at index " << i << endl;
            return;
        }
    }
    cout << "Element not found!\n";
}

// 6. Print addresses
void printAddresses(int arr[], int n) {
    cout << "Addresses of array elements:\n";
    for (int i = 0; i < n; i++)
        cout << "&arr[" << i << "] = " << &arr[i] << endl;
}

// 7. Row-major / Column-major check
void check2DArrayStorage() {
    int arr[2][3] = { {1,2,3}, {4,5,6} };
    cout << "2D Array address check:\n";
    for (int i = 0; i < 2; i++) {
        for (int j = 0; j < 3; j++) {
            cout << "&arr[" << i << "][" << j << "] = " << &arr[i][j] << "  ";
        }
        cout << endl;
    }
    cout << "\n(If row elements are consecutive → Row-major. If column elements are consecutive → Column-major)\n";
}
```

```
// Main driver
int main() {
    int arr[MAX], n;

    populateArray(arr, n);
    traverseArray(arr, n);

    insertAtIndex(arr, n, 2, 99);   // insert 99 at index 2
    traverseArray(arr, n);

    deleteAtIndex(arr, n, 3);       // delete element at index 3
    traverseArray(arr, n);

    searchElement(arr, n, 99);
    printAddresses(arr, n);

    check2DArrayStorage();

    return 0;
}
```

Q2->

```
#include <iostream>
#include <chrono>   // for measuring time
using namespace std;
using namespace std::chrono;

// Linear Search function
int linearSearch(int arr[], int n, int key) {
    for (int i = 0; i < n; i++) {
        if (arr[i] == key)
            return i;   // return index if found
    }
    return -1; // not found
}

int main() {
    // Step 1: Define and populate array
    int arr[20] = {34, 7, 23, 32, 5, 62, 78, 12, 45, 89,
```

```
                    90, 11, 17, 66, 54, 31, 29, 73, 81, 10};
    int n = 20;

    cout << "Array elements: ";
    for (int i = 0; i < n; i++) cout << arr[i] << " ";
    cout << "\n";

    // Step 2: Input element to search
    int key;
    cout << "Enter element to search: ";
    cin >> key;

    // Step 3: Measure running time
    auto start = high_resolution_clock::now();

    int index = linearSearch(arr, n, key);

    auto stop = high_resolution_clock::now();
    auto duration = duration_cast<nanoseconds>(stop - start);

    // Step 4: Display result
    if (index != -1)
        cout << "Element " << key << " found at index " << index << endl;
    else
        cout << "Element " << key << " not found in the array." << endl;

    cout << "Running time of Linear Search: " << duration.count() << "
nanoseconds\n";

    return 0;
}
```

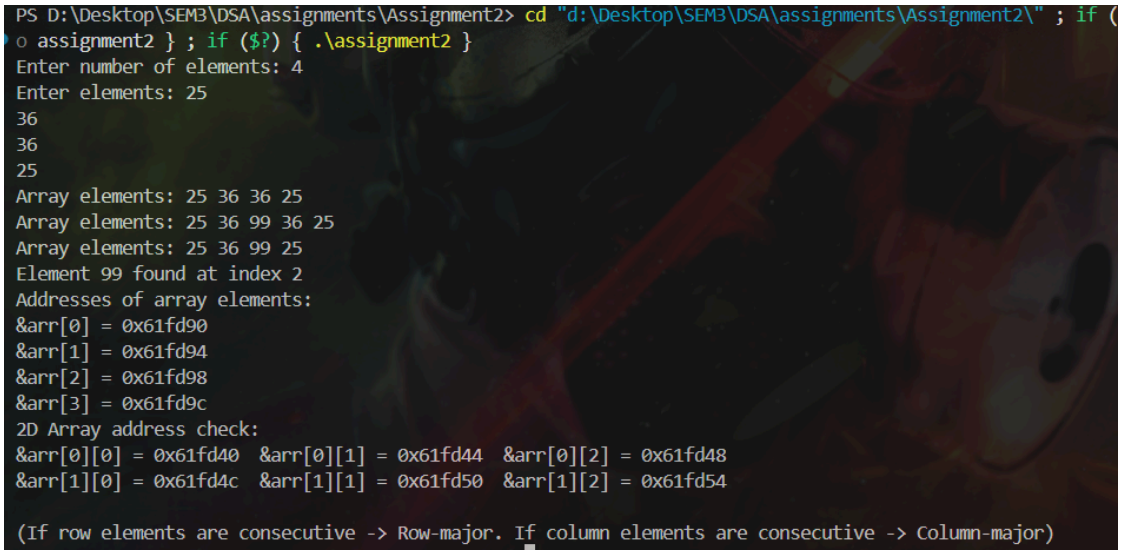
Q3->

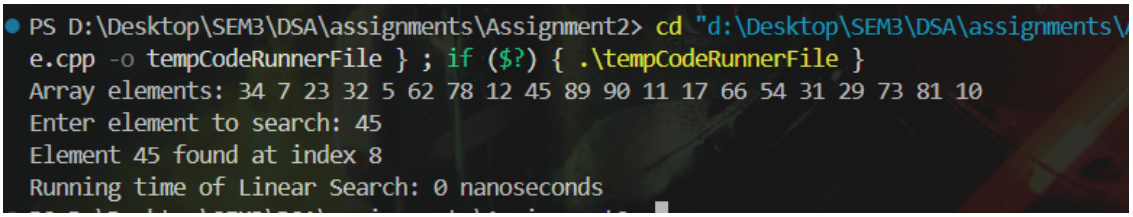
```
#include <iostream>
#include <chrono>   // for timing
using namespace std;
using namespace std::chrono;

// Iterative Binary Search
```

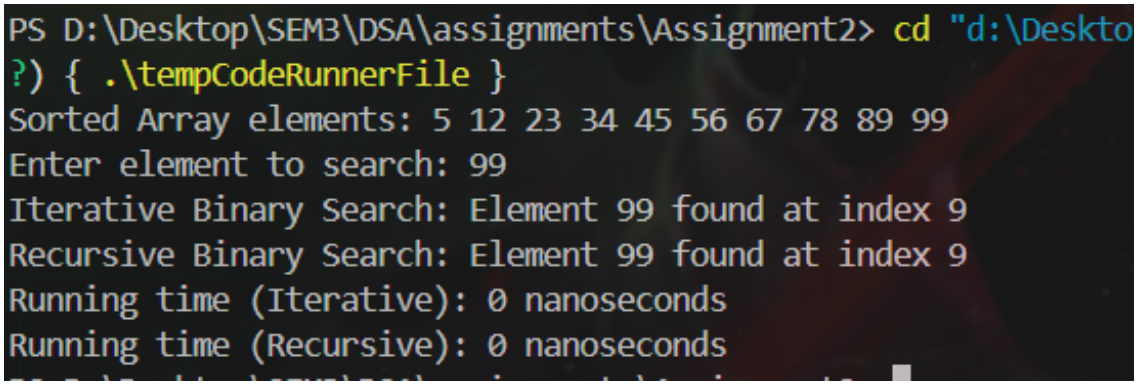
```cpp
int binarySearchIterative(int arr[], int n, int key) {
    int low = 0, high = n - 1;
    while (low <= high) {
        int mid = (low + high) / 2;
        if (arr[mid] == key)
            return mid;
        else if (arr[mid] < key)
            low = mid + 1;
        else
            high = mid - 1;
    }
    return -1;
}

// Recursive Binary Search
int binarySearchRecursive(int arr[], int low, int high, int key) {
    if (low > high) return -1;
    int mid = (low + high) / 2;
    if (arr[mid] == key)
        return mid;
    else if (arr[mid] < key)
        return binarySearchRecursive(arr, mid + 1, high, key);
    else
        return binarySearchRecursive(arr, low, mid - 1, key);
}

int main() {
    // Step 1: Define sorted array
    int arr[10] = {5, 12, 23, 34, 45, 56, 67, 78, 89, 99};
    int n = 10;

    cout << "Sorted Array elements: ";
    for (int i = 0; i < n; i++) cout << arr[i] << " ";
    cout << "\n";

    int key;
    cout << "Enter element to search: ";
    cin >> key;
```

```
    // Step 2: Iterative Search Timing
    auto start1 = high_resolution_clock::now();
    int index1 = binarySearchIterative(arr, n, key);
    auto stop1 = high_resolution_clock::now();
    auto duration1 = duration_cast<nanoseconds>(stop1 - start1);

    // Step 3: Recursive Search Timing
    auto start2 = high_resolution_clock::now();
    int index2 = binarySearchRecursive(arr, 0, n - 1, key);
    auto stop2 = high_resolution_clock::now();
    auto duration2 = duration_cast<nanoseconds>(stop2 - start2);

    // Step 4: Display results
    if (index1 != -1)
        cout << "Iterative Binary Search: Element " << key << " found at index " <<
index1 << endl;
    else
        cout << "Iterative Binary Search: Element " << key << " not found\n";

    if (index2 != -1)
        cout << "Recursive Binary Search: Element " << key << " found at index " <<
index2 << endl;
    else
        cout << "Recursive Binary Search: Element " << key << " not found\n";

    cout << "Running time (Iterative): " << duration1.count() << " nanoseconds\n";
    cout << "Running time (Recursive): " << duration2.count() << " nanoseconds\n";

    return 0;
}
```

Output -


PS D:\Desktop\SEM3\DSA\assignments\Assignment2> cd "d:\Desktop\SEM3\DSA\assignments\Assignment2\" ; if (
o assignment2 } ; if ($?) { .\assignment2 }
Enter number of elements: 4
Enter elements: 25
36
36
25
Array elements: 25 36 36 25
Array elements: 25 36 99 36 25
Array elements: 25 36 99 25
Element 99 found at index 2
Addresses of array elements:
&arr[0] = 0x61fd90
&arr[1] = 0x61fd94
&arr[2] = 0x61fd98
&arr[3] = 0x61fd9c
2D Array address check:
&arr[0][0] = 0x61fd40 &arr[0][1] = 0x61fd44 &arr[0][2] = 0x61fd48
&arr[1][0] = 0x61fd4c &arr[1][1] = 0x61fd50 &arr[1][2] = 0x61fd54
(If row elements are consecutive -> Row-major. If column elements are consecutive -> Column-major)


PS D:\Desktop\SEM3\DSA\assignments\Assignment2> cd "d:\Desktop\SEM3\DSA\assignments\
e.cpp -o tempCodeRunnerFile } ; if ($?) { .\tempCodeRunnerFile }
Array elements: 34 7 23 32 5 62 78 12 45 89 90 11 17 66 54 31 29 73 81 10
Enter element to search: 45
Element 45 found at index 8
Running time of Linear Search: 0 nanoseconds


PS D:\Desktop\SEM3\DSA\assignments\Assignment2> cd "d:\Deskto
?) { .\tempCodeRunnerFile }
Sorted Array elements: 5 12 23 34 45 56 67 78 89 99
Enter element to search: 99
Iterative Binary Search: Element 99 found at index 9
Recursive Binary Search: Element 99 found at index 9
Running time (Iterative): 0 nanoseconds
Running time (Recursive): 0 nanoseconds