Marwadi University	Marwadi University Faculty of Technology Department of Information and Communication Technology	
Subject: DAA (01CT0512)	AIM: Implementing the Searching Algorithms and understanding the time and space complexities	
Experiment No: 2	Date:	Enrolment No: 92200133002

1 Linear Search:

Theory:

Linear search is a simple searching algorithm that sequentially checks each element in a list or array until it finds the target element or reaches the end of the list. It is one of the most basic algorithms used for searching and works effectively for small datasets or when the list is unsorted.

Algorithm: [Handwritten]

Programming Language: [C++]

Code:

```
#include <iostream>

using namespace std;


int linearSearch(int arr[], int n, int x) {
    for (int i = 0; i < n; i++) {
        if (arr[i] == x)
            return i;
```

```
    }
    return -1;
}

int main() {
    int arr[] = {5, 3, 7, 8, 2};  // Example array
    int n = sizeof(arr) / sizeof(arr[0]);  // Size of the array
    int x = 7;  // Element to search for

    int result = linearSearch(arr, n, x);

    if (result == -1)
        cout << "Element not found in the array" << endl;
    else
        cout << "Element found at index " << result << endl;

    return 0;
}
```

Output:

```
Element found at index 2
PS C:\Users\Harsh\OneDrive\Documents\VS Codes\C++\DAA Codes\Sorting>
```

Marwadi University	**Marwadi University** **Faculty of Technology** **Department of Information and Communication Technology**	
Subject: DAA (01CT0512)	**AIM: Implementing the Searching Algorithms and understanding the time and space complexities**	
Experiment No: 2	**Date:**	**Enrolment No: 92200133002**

Space complexity: _____

Justification:_____

_____

Time complexity:

Best case time complexity: _____

Justification:_____

_____

Worst case time complexity: _____

Justification:_____

_____

Marwadi University	Marwadi University Faculty of Technology Department of Information and Communication Technology	
Subject: DAA (01CT0512)	AIM: Implementing the Searching Algorithms and understanding the time and space complexities	
Experiment No: 2	Date:	Enrolment No: 92200133002

2. Binary Search

Theory: Binary search is an efficient searching algorithm that works on sorted arrays or lists. It repeatedly divides the search space in half by comparing the target element with the middle element of the list. Based on the comparison, the search continues in either the left or right half of the list, narrowing down the possible positions of the target element until it is found or the search space is reduced to zero.

Binary search is much faster than linear search for large datasets due to its logarithmic time complexity.

Algorithm: [Handwritten]

Programming Language: [C++]

Code:

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

```
int binarySearch(int arr[], int l, int r, int x) {
   while (l <= r) {
      int mid = l + (r - l) / 2;

      if (arr[mid] == x)
         return mid;

      if (arr[mid] < x)
         l = mid + 1;

      else
         r = mid - 1;
   }
   return -1;
}
int main() {
   int arr[] = {2, 3, 4, 10, 40};
   int n = sizeof(arr) / sizeof(arr[0]);
   int x = 10;

   int result = binarySearch(arr, 0, n-1, x);
   if (result == -1)
      cout << "Element not found in Binary Search" << endl;
```

```
    else
        cout << "Element found at index " << result << " in Binary Search" << endl;
    return 0;
}
```

Output:

```
PS C:\Users\Harsh\OneDrive\Documents\VS Codes> cd "c:\Users\Harsh\OneD
rch.cpp -o binary_search } ; if ($?) { .\binary_search }
Element found at index 3 in Binary Search
PS C:\Users\Harsh\OneDrive\Documents\VS Codes\C++\DAA Codes\Sorting>
```

Space complexity: _____

Justification:_____

_____

Time complexity:

Best case time complexity: _____

Justification:_____

_____

Worst case time complexity: _____

Justification:_____

_____