Practical - 4

Aim:

Implement a function of Sequential Search & Binary Search and count the steps executed by function on various inputs for best case and worst case. Also write complexity in each case and draw a comparative chart.

Code:

```
#include <stdio.h>
#include <stdlib.h>

int count_linear_search = 0;
int count_binary_search = 0;

int binary_search(int arr[], int size, int target) {
    int left = 0;
    int right = size - 1;

    while (left <= right) {
        count_binary_search++;
        int mid = left + (right - left) / 2;

        if (arr[mid] == target) {
            return mid; // Target found at index mid
        }
        if (arr[mid] < target) {
            left = mid + 1; // Search in the right half
        } else {
            right = mid - 1; // Search in the left half
        }
    }
    return -1; // Target not found
}
```

```
int linear_search(int arr[], int size, int target) {
    for (int i = 0; i < size; i++) {
        count_linear_search++;
        if (arr[i] == target) {
            return i; // Target found at index i
        }
    }
    return -1; // Target not found
}

int main() {
    FILE *fp;
    fp = fopen("arr.txt", "r");
    int size = 0;
    int target = 0;

    fscanf(fp, "%d", &size);
    fscanf(fp, "%d", &target);

    int* array = (int*)malloc(size * sizeof(int));
    for (int i = 0; i < size; i++) {
        fscanf(fp, "%d", &array[i]);
    }
    fclose(fp);
    // Linear Search
    int result = linear_search(array, size, target);
    if (result != -1) {
        printf("Linear Search: Element found at index %d\n", result);
    } else {
        printf("Linear Search: Element not found\n");
    }
    printf("Size of Array used for Binary Search: %d\n", size);
    printf("Number of Steps in Linear Search: %d\n", count_linear_search);
```

```
    free(array);

    // Binary Search
    fp = fopen("arr_sorted.txt", "r");

    fscanf(fp, "%d", &size);
    fscanf(fp, "%d", &target);

    array = (int*)malloc(size * sizeof(int));
    for (int i = 0; i < size; i++) {
        fscanf(fp, "%d", &array[i]);
    }
    fclose(fp);
    result = binary_search(array, size, target);
    if (result != -1) {
        printf("\nBinary Search: Element found at index %d\n", result);
    } else {
        printf("\nBinary Search: Element not found\n");
    }
    printf("Size of Array used for Binary Search: %d\n", size);
    printf("Number of Steps in Binary Search: %d\n", count_binary_search);
    free(array);
    return 0;
}
```

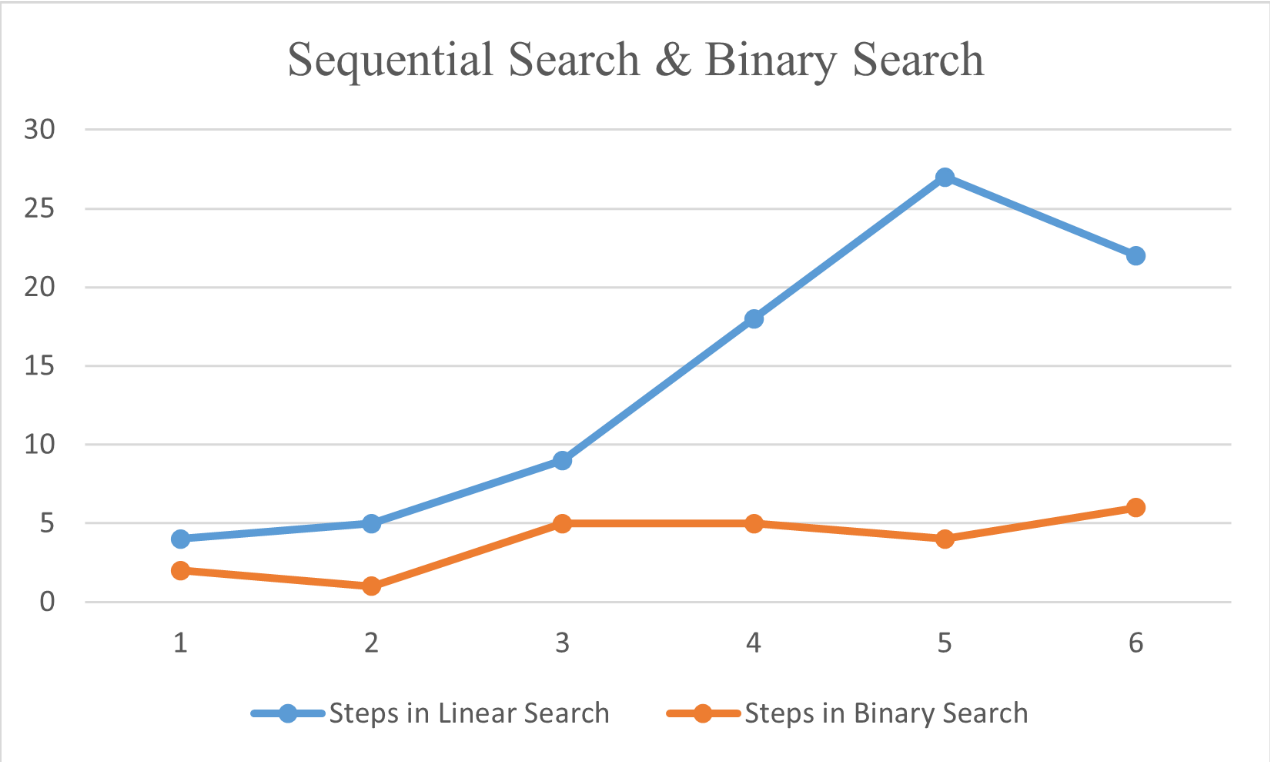
Output:

```
PS C:\Users\Lenovo\Desktop\Sem4\DAA\Prac4> .\search.exe
Linear Search: Element found at index 21
Size of Array used for Binary Search: 50
Number of Steps in Linear Search: 22

Binary Search: Element found at index 21
Size of Array used for Binary Search: 50
Number of Steps in Binary Search: 6
```

Analysis:

Value of N	Index Of Key	Steps in Linear Search	Steps in Binary Search
5	3	4	2
10	4	5	1
20	8	9	5
30	17	18	5
40	26	27	4
50	21	22	6



Conclusion:

- The program contrasts Sequential Search & Binary Search.
- The Sequential Search approach runs in O(n) time with O(1) space, comparing every index in Sequence.
- The Binary Search approach runs in O(n logn) time with O(1) space, Discarding half array every comparison.
- The Binary Search approach need sorted array to work.
- Step counts clearly show the Binary Search scales efficiently then Sequential Search.
- From this we can conclude if our data is sorted then using Binary Search is always better. Else Sequential Search is used for unsorted data.