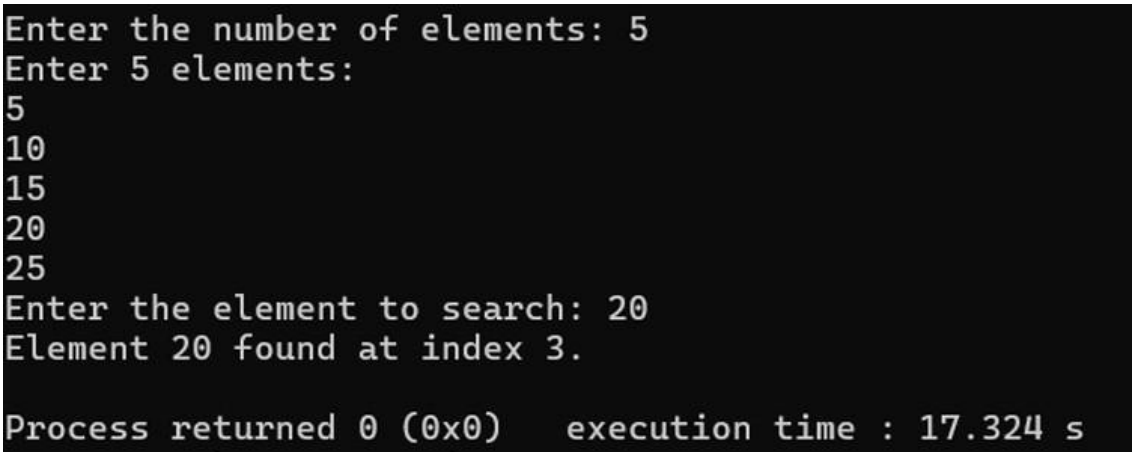
Assignment No - 3

1. Linear Search Program :

```
#include <stdio.h>

 int linearSearch(int arr[], int n, int target)
{ for (int i = 0; i < n; i++) { if (arr[i] == target)
{ return i;
      }
   }
   return -1;
}


int main() { int
   n, target;
   int arr[100]; // Fixed-size array


   printf("Enter number of elements (max 100): ");
   scanf("%d", &n);


   if (n > 100) { printf("Array size too
      large!\n"); return 1;
   }


   printf("Enter %d elements:\n", n);
   for (int i = 0; i < n; i++) {
      scanf("%d", &arr[i]);
```

```
    }

    printf("Enter element to search: "); scanf("%d",
    &target);

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

    if (result != -1)
        printf("Element found at index %d\n", result);
    else
        printf("Element not found\n");

    return 0;
}
```

Output:



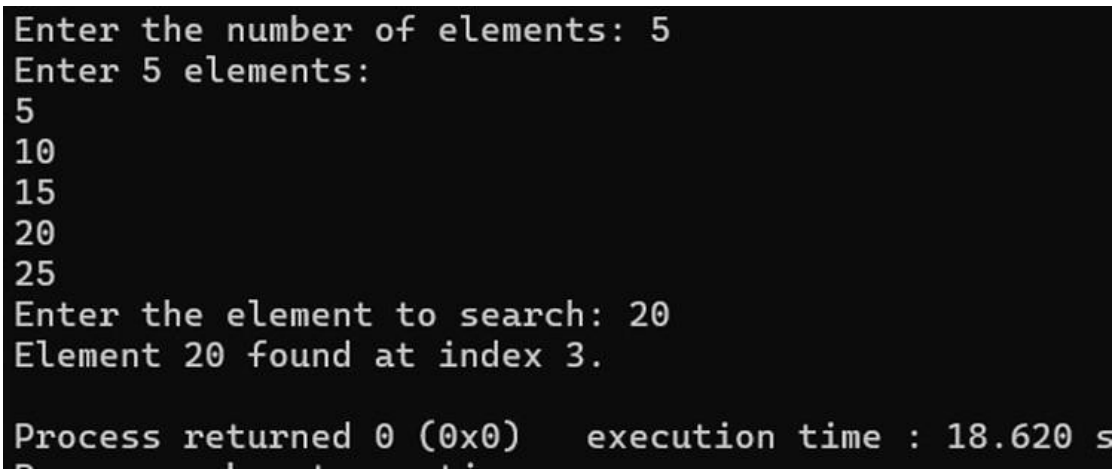
2. Binary Search (Iterative Method)

Program:

```
#include <stdio.h>

int binarySearch(int arr[], int n, int target) { int
   low = 0, high = n - 1, mid;

   while (low <= high) {
      mid = (low + high) / 2;

      if (arr[mid] == target) return
         mid;
      else if (arr[mid] < target) low
         = mid + 1;
      else
         high = mid - 1;
   }

   return -1;
}

int  main()  {  int
   n, target, i; int
   arr[100];
```

```
    printf("Enter number of elements (sorted): ");
    scanf("%d", &n);
    printf("Enter %d sorted elements:\n", n);
    for (i = 0; i < n; i++) {
        scanf("%d", &arr[i]);
    }

    printf("Enter element to search: "); scanf("%d",
    &target);

    int result = binarySearch(arr, n, target);

    if (result != -1)
        printf("Element found at index %d\n", result);
    else
        printf("Element not found\n");

    return 0;
}
```

Output:

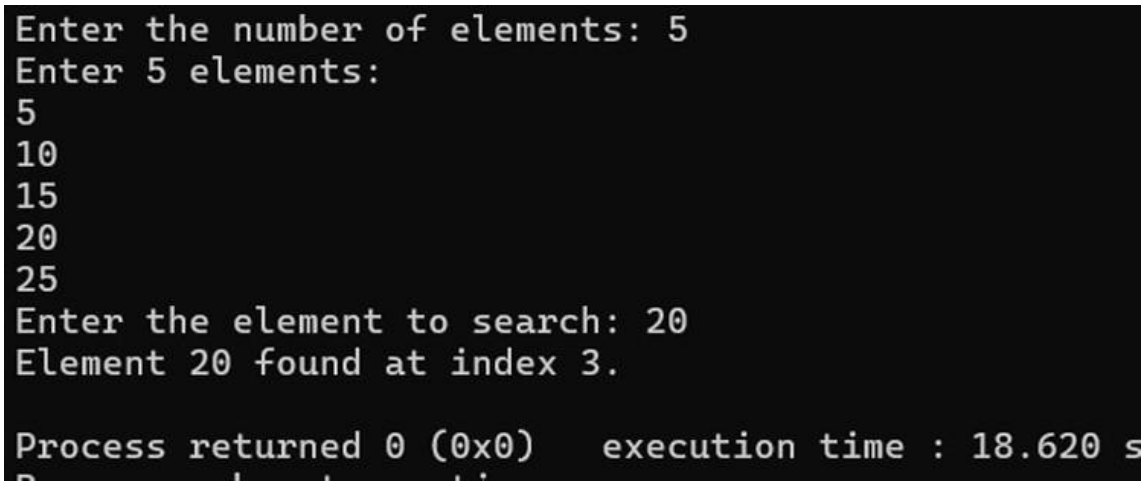


(Recursive Method) Program:

```
#include <stdio.h>

int binarySearch(int arr[], int low, int high, int target) {
   int mid; if (low <= high) { mid = (low + high) / 2; if
   (arr[mid] == target) return mid; else if (arr[mid] >
   target) return binarySearch(arr, low, mid - 1, target);
   else
         return binarySearch(arr, mid + 1, high, target);
   }
   return -1;
}

int main() {
   int arr[100], n, target, i, result;

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

   printf("Enter %d sorted elements:\n", n);
   for (i = 0; i < n; i++) {
      scanf("%d", &arr[i]);
   }

   printf("Enter element to search: "); scanf("%d",
   &target);
```

```
    result = binarySearch(arr, 0, n - 1, target);

    if (result != -1)
        printf("Element found at index %d\n", result);
    else
        printf("Element not found\n");

    return 0;
}
```

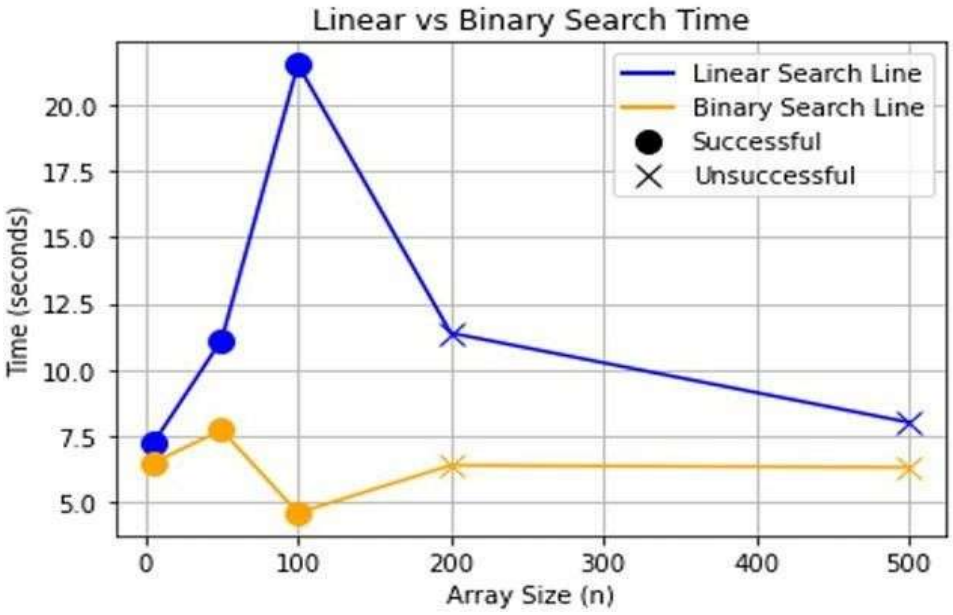
Output:



Input:

Array size (n)	key value (s)	Linear Time (s)	Binary Time (s)	search	
5	500	7.219 s	6.495 s	Successful	
50	912	11.136 s	7.739 s	Successful	
100	35	21.506 s	4.615 s	Successful	
200	44	11.392 s	6.407 s	Unsuccessful	
500	1000	8.027 s	6.329 s	Unsuccessful	

Graph of Linear Search VS Binary Search (Recursive Method):



Conclusion:

Binary search is consistently faster than linear search in both successful and unsuccessful cases.

Linear search takes more time when the key is not found (unsuccessful), as it checks every element.

Binary search has less variation in time between successful and unsuccessful searches because it always halves the array. For large and sorted arrays, binary search is highly efficient, especially when the key might not be present.