Exp 1

PAGE No.
DATE 31/1/23

Name: Shreerang Mhatre
Rollno: 29
Batch: A2

Q) write a program to search a number from the given list of numbers stored in an array using linear search and binary search

Output for linear search -

```
>> gcc .\linear1.c
>> .\a.exe

How many elements? 10
Enter array elements: 10
20
30
40
50
60
70
80
90
55
Enter element to search: 60
Element found at index 5
```

31/1/23

Exp 1

PAGE NO.
DATE 9/2/23

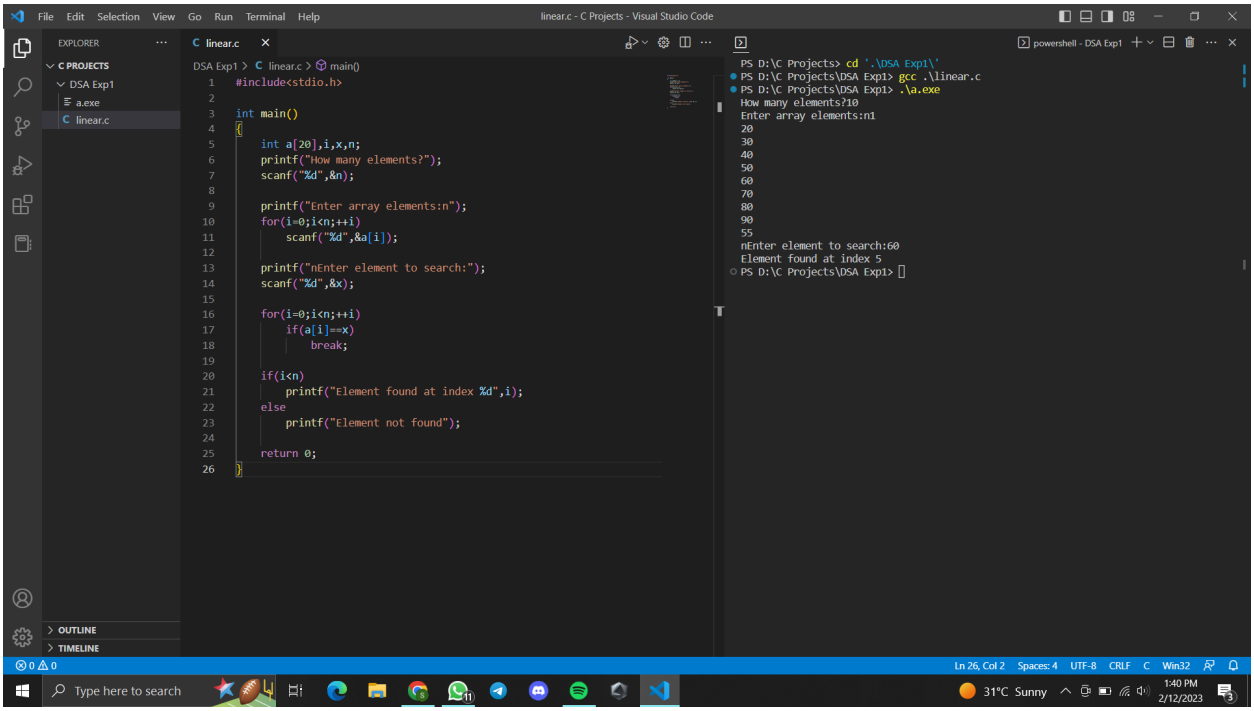
Name: Shreerang Mhatre
Roll no: 29
Batch: A2

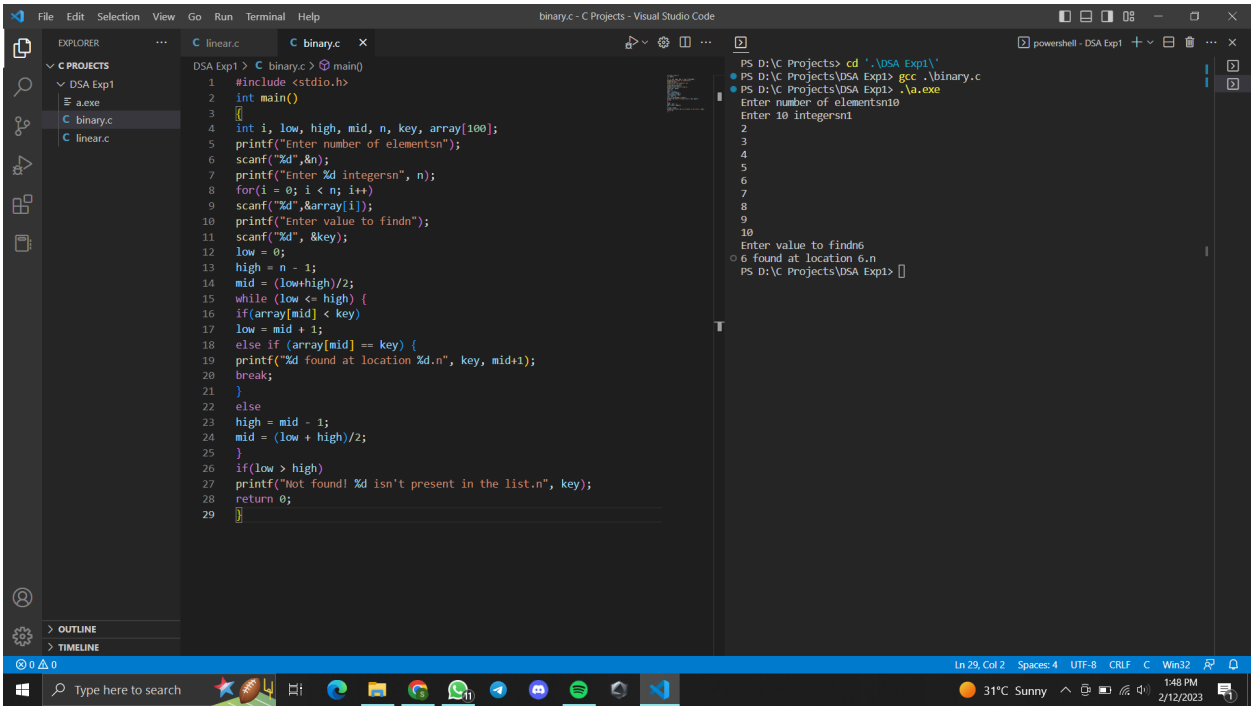
Q) write a program to search a number from the given list of numbers stored in an array using linear search and binary search.

Output for Binary search -

>> gcc .\ binary
>> .\ a.exe

Enter number of elements: 10
Enter 10 integers: 1
2
3
4
5
6
7
8
9
10
Enter value to find: 6
6 found at location 5





Linear Search:

```
#include<stdio.h>

int main()
{
   int a[20],i,x,n;
   printf("How many elements?");
   scanf("%d",&n);

   printf("Enter array elements:n");
   for(i=0;i<n;++i)
      scanf("%d",&a[i]);

   printf("nEnter element to search:");
   scanf("%d",&x);

   for(i=0;i<n;++i)
      if(a[i]==x)
         break;

   if(i<n)
      printf("Element found at index %d",i);
   else
      printf("Element not found");

   return 0;
}
```

Binary Search:

```
#include <stdio.h>
int main()
{
int i, low, high, mid, n, key, array[100];
printf("Enter number of elementsn");
scanf("%d",&n);
printf("Enter %d integersn", n);
for(i = 0; i < n; i++)
scanf("%d",&array[i]);
printf("Enter value to findn");
scanf("%d", &key);
low = 0;
high = n - 1;
mid = (low+high)/2;
while (low <= high) {
if(array[mid] < key)
low = mid + 1;
else if (array[mid] == key) {
printf("%d found at location %d.n", key, mid+1);
break;
}
else
high = mid - 1;
mid = (low + high)/2;
}
if(low > high)
printf("Not found! %d isn't present in the list.n", key);
return 0;
}
```

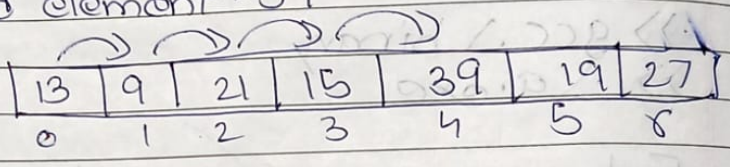
PAGE No.
DATE / /

Theory

1) Linear search -

Linear search, often known as sequential search, is the most basic technique. In this type of search, you go through the entire list and try to fetch a match for a single element. If you find a match.

Searched element - 39



step1: The searched element 39 is compared to the first element of an array, which is 13.
The match is not found, now move on the next element and try to implement a comparison.

step2: Now, search element 39 is compared to the second element of an array, 9.

13 9 21 15 39 19 27
~~1~~ 1V
39

step3: Now, search element 39 is compared with the third element, which is 21

13 9 21 15 39 19 27
↑
39

Again, both the elements are not matching, you move onto the next following element.

step4: Next, search element 39 is compared with the fourth element, which is 15.

13 9 21 15 39 19 27
↑
39

As both are not matching, you move on to the next element.

step5: Next, search element 39 is compared with the fifth element 39.

13 9 21 15 39 19 27
↑
39

A perfect match is found, you stop comparing any further elements & terminate the Linear search Algorithm & display the element found at location 4.

2) Binary Search

Binary search is a searching algorithm for finding an element's position in a sorted array. In this approach, the element is always searched in the middle of a portion of an array. Binary search can be implemented only on a sorted list of items. If the elements are not sorted already, we need to sort them first.

step 1) The array in which searching is to be performed is -

3	4	5	6	7	8	9

Let x = 4 be the element to be searched

step 2) Set two pointers low and high at the lowest and the highest positions respectively.

3 4 5 6 7 8 9

low ↵ ↳ high

step 3) Find the middle [mid] of the array i.e arr[(low + high)/2] = 6

3 4 5 6 7 8 9

↑

mid

→

step 4> If x == mid, then return mid. Else, compare the element to be searched with m.

step 5> If x > mid, compare x with the middle element of the elements on the right side of mid. This is done by setting low to low = mid + 1.

step 6> Else, compare x with the middle element of the elements on the left side of mid. This is done by setting high to high = mid - 1.

3 4 5 6 7 8 9
low ↑ high ↑

step 7> Repeat steps 3 to 6 until low meets high

3 4 5
↑ mid

step 8> x = 4 is found.

3 4 5
↓
x = mid

Algorithm
write respective algorithm or pseudo code for linear search & binary search

* Pseudocode for Linear Search -

step 1) start
Step 2) linear_search (Array, value)
step 3) For each element in the array
step 4) If (searched element == value)
step 5) Returns the searched element location
step 6) end if
step 7) end for
step 8) end.

* Pseudo code for Binary Search -

step 1) start
step 2) Input sorted array in "a[]" and element to be searched in "x" and size of array in "size"
Step 3) Initialize low = 0, high = size - 1
step 4) Repeat until low >= high
: mid = (low + high) / 2
: If a[mid] is equal to x then print index value of mid & Goto step 6
: Else
If a[mid] < x
low = mid + 1
else high = mid - 1
step 5) Print x not found in the list
step 6) Stop

* FAQs:

Q1) What are the applications of searching?

→ There are numerous searching algorithms in a data structure such as linear search, binary search, interpolation search, jump search, exponential search, Fibonacci search and recursive program to search an element linearly in the given array.

Q2) Compare and contrast linear search and binary search?

→

Linear search	Binary search
① The linear search starts searching from the first element and compares each element with a searched element till the element is not found	① It finds the position of the searched element by finding the middle element of the array.
② In a linear search, the elements don't need to be arranged in sorted order	② The pre-condition for the binary search is that the elements must be arranged in a sorted order.
③ The linear search can be implemented on any linear data structure such as an array, linked list, etc.	③ The implementation of binary search is limited as it can be implemented only on those data structures that have two-way traversal.