**ASSIGNMENT - 2**

**1.PROBLEM STATEMENT**

Write a program in C to search an element from a list. Give option to the user to perform linear search or binary search

**2.ALGORITHM**

The program presents an option to the user to perform either linear search or binary search.

**Algorithm Linear\_Search**

**Input:** A pointer to the array named arr[1…n] with size n containing the list of integers and the integer to be searched is called item.

**Output:** If the given integer is found , the index of the position of the element is returned with a successful message , else an unsuccessful message is shown.

**Remarks:** It is assumed that the array is not empty.

**Steps:**

1. NOTFOUND=-1 *// Invalid index to indicate unsuccessful search*
2. flag = 0 *//* *To indicate whether element was found or not*
3. **For** i=1 to n **do** *//**Traversing the whole list*
4. **If**(arr[i]=item) **then** *//**if value at index equals item*
5. flag=1 *// change value of flag to 1*
6. return i *// index of the element is returned*
7. **EndIf**
8. **If**(flag=0) **then** *// if flag value is not changed*
9. return NOTFOUND*//**return invalid index to indicate absent item*
10. **EndIf**
11. **Stop**

**Algorithm Binary\_Search**

**Input:** A pointer to the integer array named arr[1...n] with size n containing the list of integers and the integer to be searched is called item.

**Output:** If the given integer is found , the index of the position of the element is returned with a successful message , else an unsuccessful message is shown.

**Remarks:** It is assumed that the array is not empty and is sorted in ascending order.

**Steps:**

1. low=1 , high=n
2. flag=FALSE
3. **While**(low≤high) **do**
4. mid = (low+high)/2
5. **If**(arr[mid]=item) **then**
6. flag=TRUE
7. return mid
8. **EndIf**
9. **If**(arr[mid]<item) **then**
10. low=mid+1
11. **EndIf**
12. **If**(arr[mid]>item) **then**
13. high=mid-1
14. **EndIf**
15. **EndWhile**
16. **If**(flag=FALSE) **then**
17. return NOTFOUND
18. **EndIf**
19. **Stop**

**3.SOURCE CODE**

​#​include​<​stdio.h​>

​#​include​<​stdlib.h​>

​#​define​ ​NOTFOUND​ -​1

// Function to ask the user whether to continue or to exit

​int​ ​prompt​(​void​)

​{

​        ​int​ num;

​        ​printf​(​"​\n​Press:​\n​0 To Exit​\n​1 to continue​\n​"​);

​        ​scanf​(​"​%d​"​,&num);

​        ​return​ num;

​}

// Function to take input in an array from the user

​void​ ​getarr​(​int​\*arr,​int​ num)

​{

​        ​int​ i;

​        ​for​(i=​0​;i<num;i++)

​                ​scanf​(​"​%d​"​,&arr[i]);

​}

// sorting algorithm for binarysearch function

void insertionsort(int\* arr,int num)

{

int i,j,temp;

for(i=1;i<num;i++)

{

temp=arr[i];

for(j=i-1; j>=0 && arr[j]>temp;j--)

arr[j+1]=arr[j];

arr[j+1]=temp;

}

}

// Function to display the contents of an array

​void​ ​disparr​(​int​\*arr,​int​ num)

​{

​        ​int​ i;

​        ​for​(i=​0​;i<num;i++)

​                ​printf​(​"​%d​ ​"​,arr[i]);

​}

​int​ ​binarysearch​(​int​\*arr,​int​ num,​int​ item)

​{

​        ​int​ low=​0​,high=num-​1​,mid; // setting initial lower and upper index variables

insertionsort(arr,num); // sorting the list to enable binarysearch

​        ​while​(low<=high)

​        {

​                mid=(low+high)/​2​; // calculating the middle element

​                ​if​(arr[mid]==item) // if mid element equals item

​                        ​return​ mid; // return the middle index

​                ​else​ ​if​(item<arr[mid]) //if mid element is greater than item

​                        high=mid-​1​; // set the value of high than mid-1

​                ​else​ ​if​(item>arr[mid]) // if mid element is less than item

​                        low=mid+​1​; // set the value of low to mid+1

​        }

​        ​return​ NOTFOUND; // if the above does not return,item is not found

​}

​int​ ​linearsearch​(​int​\*arr,​int​ num,​int​ item)

​{

​        ​int​ i;

​        ​for​(i=​0​;i<num;i++) //traversing the array

​                ​if​(arr[i]==item) // if an element equals item

​                        ​return​ i; // return the corresponding index

​        ​return​ NOTFOUND;// if the above does not return , item is not found

​}

​int​ ​main​(​void​)

​{

​        ​int​ num,i,ch,item,found,val;

​        ​int​\*arr; // pointer to hold an array

​while​(​1​)

​        {

printf(“To search a given list of integers: \n“);

​        ​printf​(​"​Enter the number of elements needed: ​"​);

​        ​scanf​(​"​%d​"​,&num);

​        ​if​(num<​2​) // Checking if at least two elements are present

​        {

​                ​printf​(​"​Invalid Array Length​\n​Please Retry​"​);

​                ​return​ ​0​;

​        }

​        arr=(​int​\*)​calloc​(num,​sizeof​(​int​)); // creating the array

​                ​printf​(​"​\n​Enter ​%d​ elements: ​"​,num);

​             ​getarr​(arr,num);

​

​                ​printf​(​"​\n​Enter the element to be searched: ​"​);

​                ​scanf​(​"​%d​"​,&item);

​

​                ​printf​(​"​\n\n​1.Binary Search\n​2.Linear Search​"​);

​                ​printf​(​"​\n​Enter Your Choice: ​"​);

​                ​scanf​(​"​%d​"​,&ch);

​                ​switch​(ch)

​                {

​                        ​case​ ​1​:

​                        found=​binarysearch​(arr,num,item);

​                        if​(found==NOTFOUND)

​                        printf​(​"​\n​%d​ is not present in the list​"​,item);

​                        ​else

​                        ​printf​(​"​\n​%d​ Found At ​%d​"​,item,found);

​                        val=​prompt​(); //Asking user whether to continue

​                        if​(val==​0​)

​                        ​return​ ​0​; // Program terminates if user enters

​                        ​break​;

​                        ​case​ ​2​:

​                        found=​linearsearch​(arr,num,item);

​                        ​if​(found!=NOTFOUND)

​                        ​printf​(​"​\n​%d​ found at index ​%d​"​,item,found);

​                        ​else

​                              printf​(​"​\n​%d​ not found in the array​"​,item);

​                        val=​prompt​(); // Asking user whether to continue

​                        ​if​(val==​0​)

​                     ​return​ ​0​; // Program terminates if user enters 0

​                        ​break​;

​                        ​default​:

​                 ​printf​(​"​Wrong Input-Please Enter Valid Choice​"​);

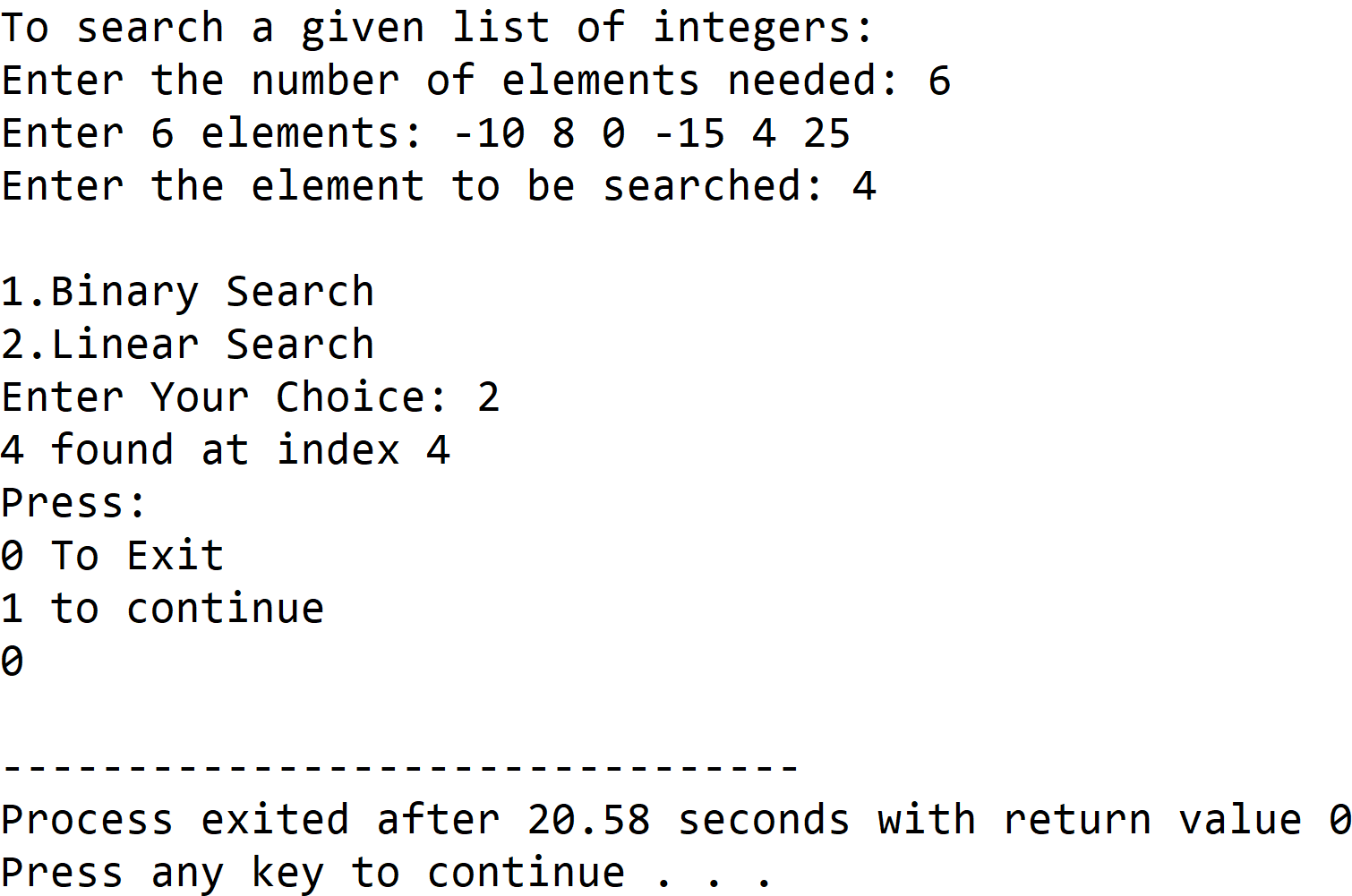
​                }

​        }

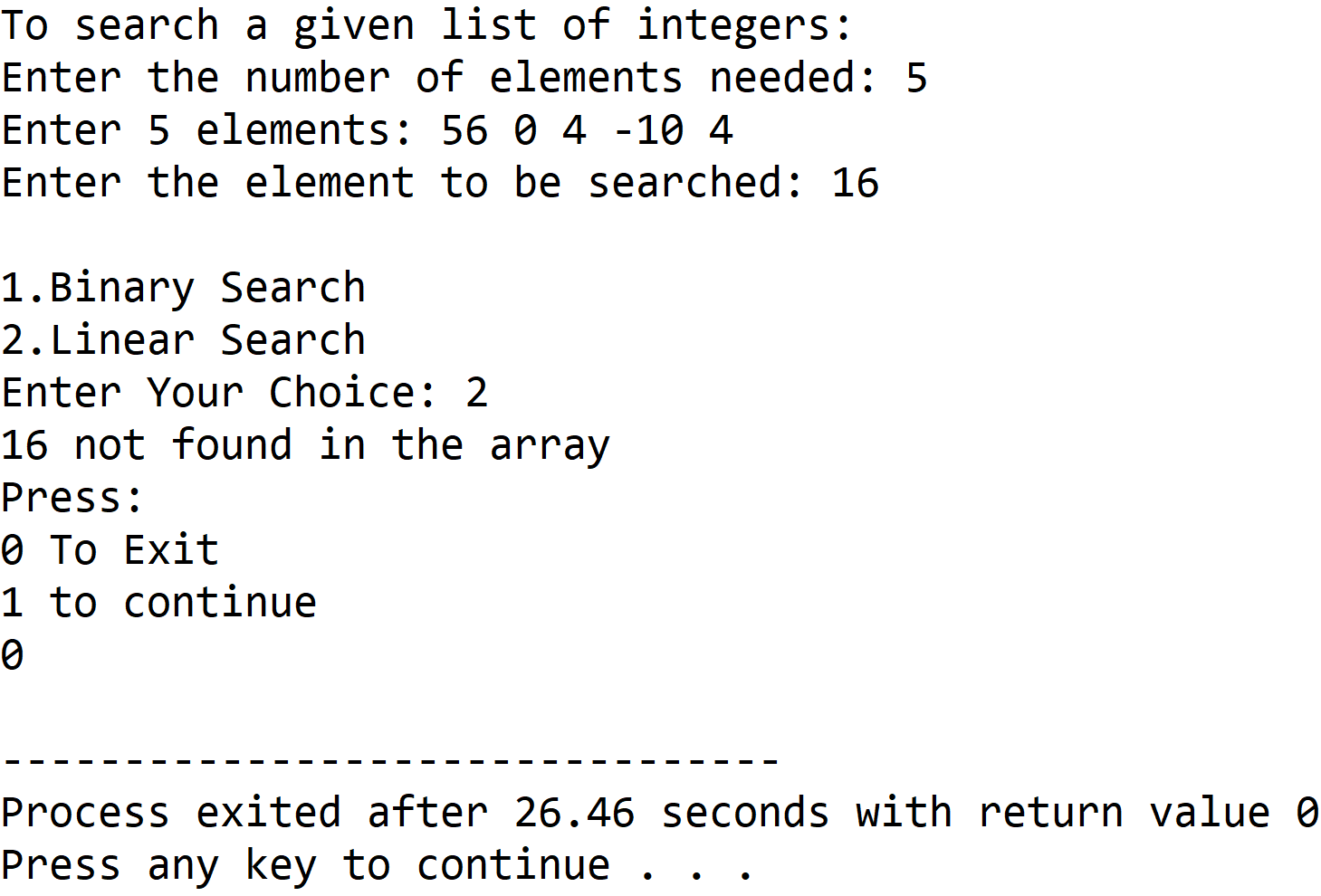
​}

**4. OUTPUT**

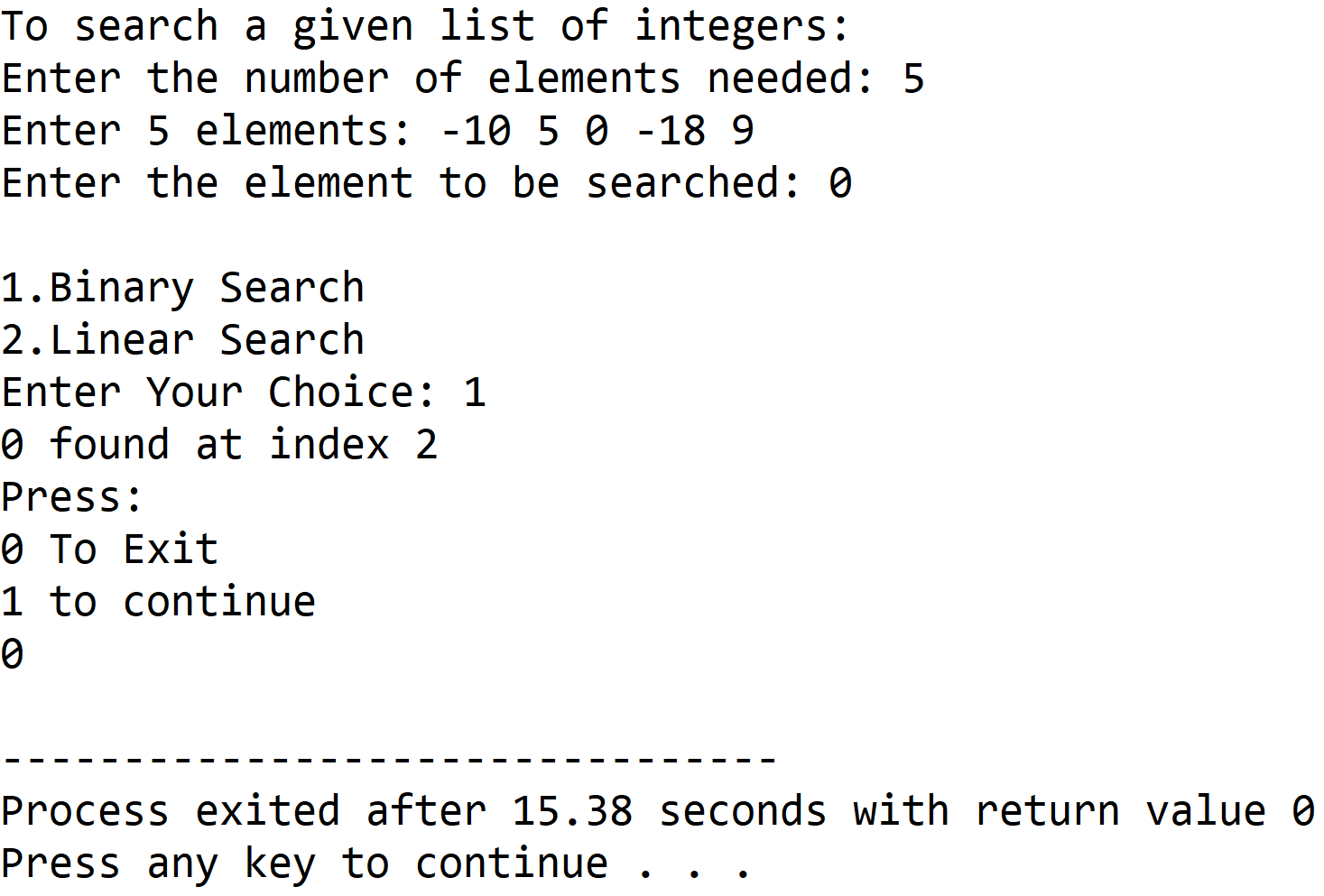
**Set 1:** Item Found in Linear Search



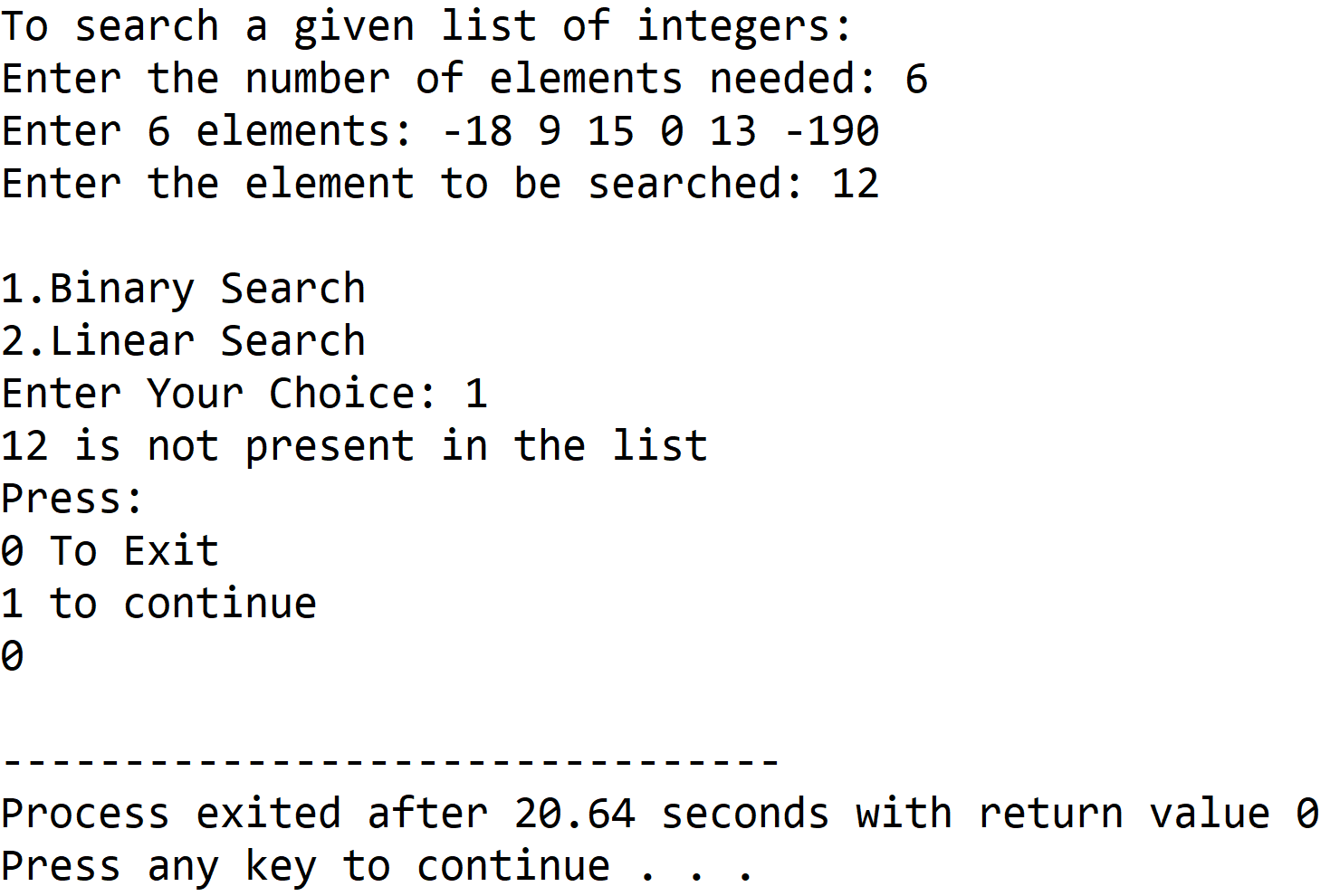
**Set 2:** Item Not Found in Linear Search



**Set 3:** Item Found in Binary Search



**Set 4:** Item Not Found in Binary Search



**5. DISCUSSION**

1. **Variable Description:**

**In Binary Search Function:**

* low : lower index value of the array under consideration
* high : upper index value of the array under consideration
* mid : index of the middle element with respect to low and high

**In Linear Search Function:**

* i : loop counter used to access the array

**In Main Function:**

* num : size of the array required by the user
* i : loop counter
* ch : holds the choice entered by the user in switch-case construct
* item: element to be found in the list
* val: stores return value of prompt function

**Macros Used:**

* NOTFOUND: holds a constant value -1 which is an invalid index value indicative of the absence of item in the given list

1. **Limitations:**

* The binarysearch function relies on having the array in sorted in ascending order, so the array entered by the user has to be sorted by using a sorting algorithm,in the above program insertion sort is used , if the data is unsorted , the binarysearch function will not work .
* The program uses an integer array to hold the list of integers entered by the user, since arrays are static data structures ,their size cannot be manipulated once it is allocated in the memory.

1. **Uses:**

* The above program can be used to search an integer value from any database consisting of a list of integers. For example, it can be used by educational institutions to enable automation of searching of roll numbers of students.

1. **Future Scope:**

* The list of integers can be stored in a linked list , enabling more elasticity in manipulation of size of the list.
* A sorting algorithm with better time and space complexity can to used for a more efficient program.