Assignment 10

Dated __________

Problem Statement

Write a C program to search for a particular element in a sorted array using the **Binary Search** method. The program should allow the user to input a sorted array of elements and then search for a specific element using an efficient divide-and-conquer approach.

---

2. Input & Output Description

- **Input:**
    - The number of elements in the array (n), where n≤100n \leq 100n≤100.
    - The sorted elements of the array.
    - The element to be searched (key).
- **Output:**
    - If the element is found, display its position.
    - If the element is not found, display a message indicating that the element is not present in the array.

---

3. Algorithm for Binary Search

Step 1: Take input for the number of elements (n).
Step 2: Check if n exceeds the maximum allowed size (100). If true, display an error message and terminate the program.
Step 3: Take input for n elements and store them in an **already sorted** array (A[]).
Step 4: Take input for the search element (key).
Step 5: Initialize three variables:

- low = 0 (starting index).
- high = n - 1 (ending index).
- mid = (low + high) / 2 (middle index).

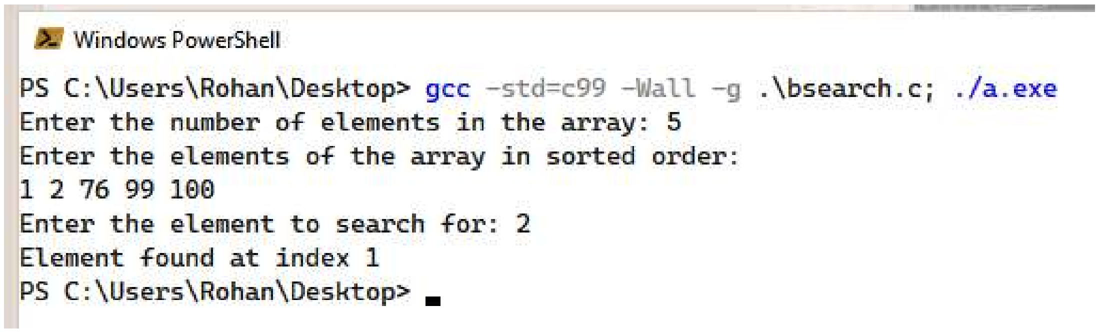
Step 6: Repeat steps 7-9 while low <= high:
Step 7: If A[mid] == key, print the position (mid+1) and terminate the search.
Step 8: If A[mid] < key, update low = mid + 1 (search right half).

Step 9: If A[mid] > key, update high = mid - 1 (search left half).
Step 10: If the loop exits without finding the element, display "Element not found" message.
Step 11: End.

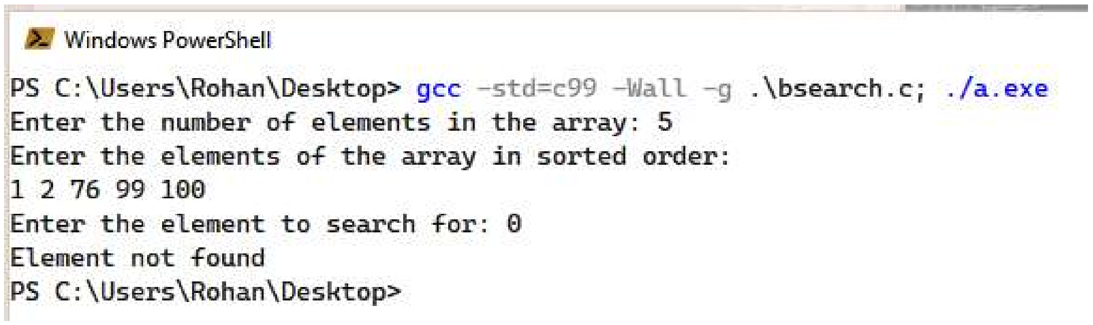
Source Code

```c
#include <stdio.h>
#define MAX 100  // Defining the maximum size of the array

// Function for performing Binary Search
void binarySearch(int A[], int n, int key) {
    int low = 0, high = n - 1, mid;

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

        if (A[mid] == key) {  // If key is found
            printf("\nElement %d found at position %d\n", key, mid + 1);
            return;
        } else if (A[mid] < key) {  // Search in the right half
            low = mid + 1;
        } else {  // Search in the left half
            high = mid - 1;
        }
    }

    printf("\nElement %d not found in the array.\n", key);
}

// Main function
int main() {
    int A[MAX], n, key, i;

    // Taking input for number of elements
    printf("Enter the number of elements: ");
    scanf("%d", &n);

    // Checking if the size exceeds MAX
    if (n > MAX) {
        printf("\nSize exceeds the limit of %d. Cannot store elements.\n", MAX);
        return 1;
    }

    // Taking input for sorted elements
    printf("Enter %d sorted elements:\n", n);
    for (i = 0; i < n; i++) {
        printf("A[%d]: ", i);
        scanf("%d", &A[i]);
    }
    // Taking input for the element to search
    printf("\nEnter the element to search: ");
    scanf("%d", &key);

    // Calling Binary Search function
    binarySearch(A, n, key);

    return 0;
}
```

Output



```
Windows PowerShell
PS C:\Users\Rohan\Desktop> gcc -std=c99 -Wall -g .\bsearch.c; ./a.exe
Enter the number of elements in the array: 5
Enter the elements of the array in sorted order:
1 2 76 99 100
Enter the element to search for: 2
Element found at index 1
PS C:\Users\Rohan\Desktop>
```



```
Windows PowerShell
PS C:\Users\Rohan\Desktop> gcc -std=c99 -Wall -g .\bsearch.c; ./a.exe
Enter the number of elements in the array: 5
Enter the elements of the array in sorted order:
1 2 76 99 100
Enter the element to search for: 0
Element not found
PS C:\Users\Rohan\Desktop>
```

Discussion

The array must be sorted for this type of search. When the element is not found, this condition should be handled appropriately.

<u>Teacher’s signature</u>