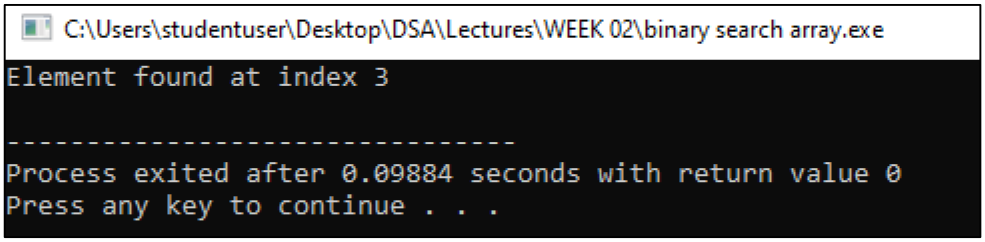
# BINARY SEARCH CODE IMPLEMENTATION

## ARRAY

```
#include <iostream>
using namespace std;

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

    while (low <= high) {
        int mid = low + (high - low) / 2;  // Calculate the middle index

        // Check if target is at mid
        if (arr[mid] == target) {
            return mid;
        }
        // If target is smaller, ignore the right half
        else if (arr[mid] > target) {
            high = mid - 1;
        }
        // If target is larger, ignore the left half
        else {
            low = mid + 1;
        }
    }
    // If we reach here, the element was not present
    return -1;
}


int main() {
    int arr[] = {2, 3, 4, 10, 40};
    int size = sizeof(arr) / sizeof(arr[0]);
    int target = 10;

    int result = binarySearch(arr, size, target);
    if (result != -1) {
        cout << "Element found at index " << result << endl;
    } else {
        cout << "Element not found" << endl;
    }

    return 0;
}
```

**OUTPUT**


C:\Users\studentuser\Desktop\DSA\Lectures\WEEK 02\binary search array.exe

```
Element found at index 3

--------------------------------
Process exited after 0.09884 seconds with return value 0
Press any key to continue . . .
```

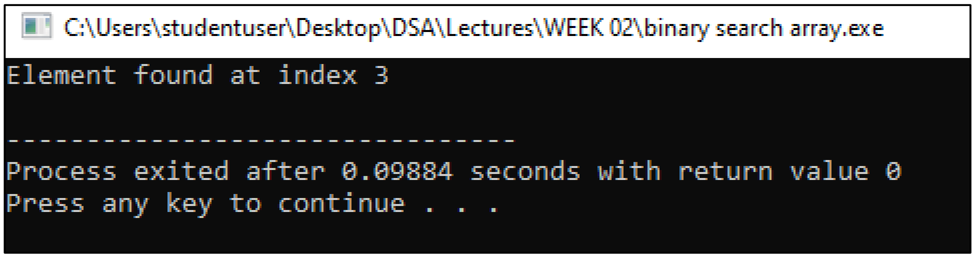
## LINKED LIST

```
#include <iostream>
using namespace std;

// Definition for singly-linked list node.
class ListNode {
public:
    int val;
    ListNode* next;

    ListNode(int x) : val(x), next(nullptr) {}
};

// Definition for LinkedList class.
class LinkedList {
public:
    ListNode* head;

    LinkedList() : head(nullptr) {}

    // Function to add a new node to the linked list.
    void insert(int new_data) {
        ListNode* new_node = new ListNode(new_data);
        new_node->next = head;
        head = new_node;
    }
```

```
// Function to find the middle of the linked list.
ListNode* findMiddle(ListNode* start, ListNode* end) {
   if (start == nullptr) return nullptr;

   ListNode* slow = start;
   ListNode* fast = start;

   while (fast != end && fast->next != end) {
      slow = slow->next;
      fast = fast->next->next;
   }

   return slow;
}

// Function to perform binary search on the linked list.
ListNode* binarySearch(int target) {
   ListNode* start = head;
   ListNode* end = nullptr;

   while (start != end) {
      // Find the middle of the current range
      ListNode* mid = findMiddle(start, end);

      if (mid == nullptr) return nullptr;

      // Check if the middle element is the target
      if (mid->val == target) {
        return mid;
      }
      // If the target is smaller, search the left half
      else if (mid->val > target) {
        end = mid;
      }
      // If the target is larger, search the right half
      else {
        start = mid->next;
      }
    }

    // If we reach here, the target is not present in the list
    return nullptr;
  }

  // Function to print the linked list (for debugging purposes)
  void printList() {
    ListNode* temp = head;
    while (temp != nullptr) {
      cout << temp->val << " -> ";
      temp = temp->next;
    }
    cout << "NULL" << endl;
  }
};
```

```
int main() {
   // Create a LinkedList object and add elements to it.
   LinkedList list;

   // Insert elements (manually adding in sorted order for simplicity)
   list.insert(9);
   list. insert(7);
   list. insert(5);
   list. insert(3);
   list. insert(1);

   // Print the list (optional)
   cout << "Linked List: ";
   list.printList();

   int target = 5;

   // Perform binary search
   ListNode* result = list.binarySearch(target);
   if (result != nullptr) {
      cout << "Element found with value " << result->val << endl;
   } else {
      cout << "Element not found" << endl;
   }

   return 0;
}
```

## OUTPUT



C:\Users\studentuser\Desktop\DSA\Lectures\WEEK 02\binary search array.exe

```
Element found at index 3

--------------------------------
Process exited after 0.09884 seconds with return value 0
Press any key to continue . . .
```