**Write a c program to find elements using array CODE:**

#include <stdio.h>

int linearSearch(int arr[], int size, int target) { for (int i = 0; i < size; i++) { if (arr[i] == target) { return i;

}

}

return -1;

}

int main() {

int arr[] = {10, 20, 30, 40, 50}; int size = sizeof(arr) / sizeof(arr[0]); int target;

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

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

if (result != -1) {

printf("Element %d found at index %d.\n", target, result);

} else {

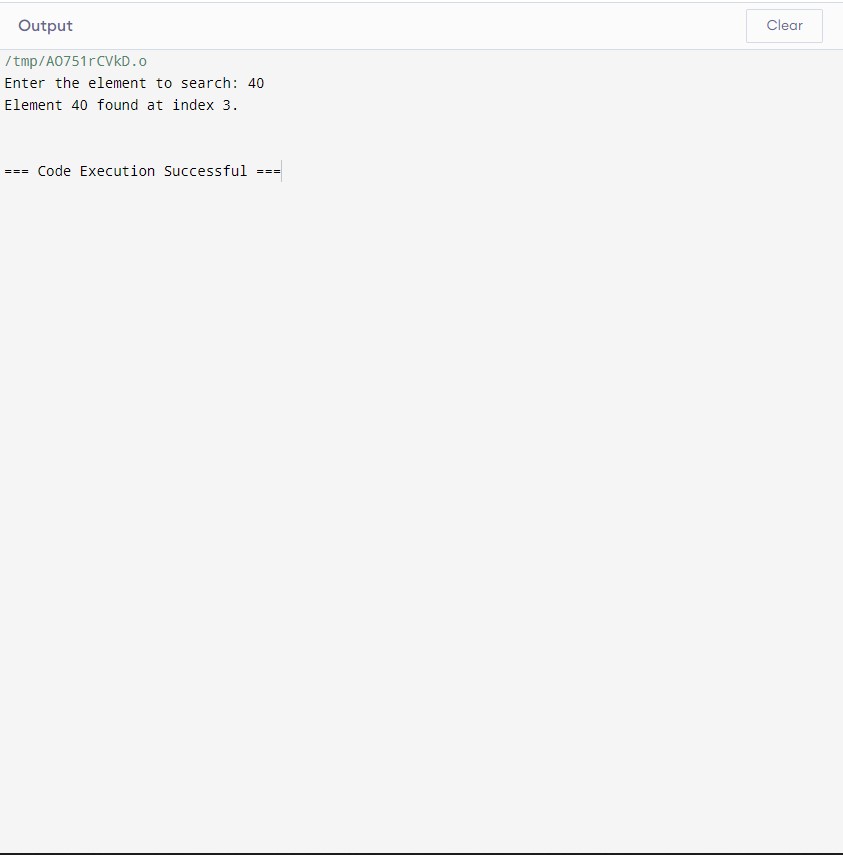
printf("Element %d not found in the array.\n", target);

}

return 0;

}

**OUTPUT:**



**TIME COMPLEXITY:**

Time Complexity**:** O(n)- (worst and average case)

 Best Case: O(1)- (if the target is the first element)

**Write a c program to find element using linklist CODE:**

#include <stdio.h> #include <stdlib.h> struct Node { int data;

struct Node\* next;

};

struct Node\* createNode(int data) {

struct Node\* newNode = (struct Node\*)malloc(sizeof(struct Node)); newNode->data = data; newNode->next = NULL; return newNode;

}

int search(struct Node\* head, int target) { struct Node\* current = head; int index = 0;

while (current != NULL) { if (current->data == target) { return index;

}

current = current->next; index++;

}

return -1;

}

int main() {

struct Node\* head = createNode(10);

head->next = createNode(20);

head->next->next = createNode(30); head->next->next->next = createNode(40); head->next->next->next->next = createNode(50);

int target;

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

int result = search(head, target);

if (result != -1) {

printf("Element %d found at index %d.\n", target, result);

} else {

printf("Element %d not found in the linked list.\n", target);

}

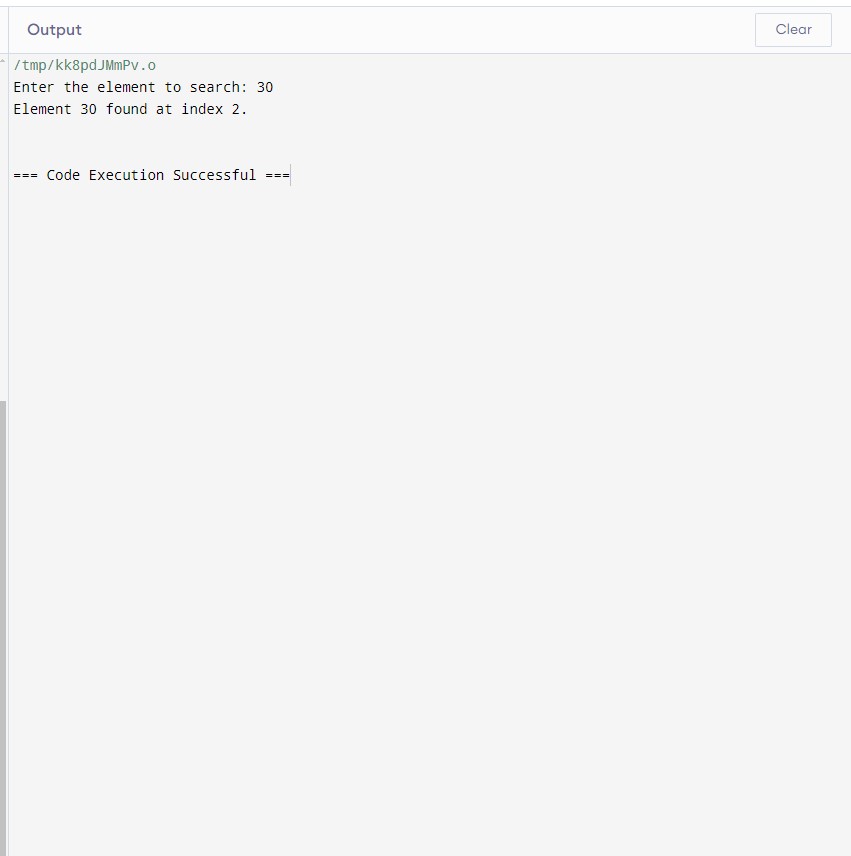
struct Node\* current = head; struct Node\* nextNode; while (current != NULL) { nextNode = current->next; free(current); current = nextNode;

}

return 0;

}

**OUTPUT:**



**TIME COMPLEXITY:**

 Time Complexity: O(n)-(worst and average case)

 Best Case: O(1)- (if the target is the first element)

**Conclusion:**

* **For Search Operations:** If you frequently need to search for elements and the size of the data set is relatively static, arrays are typically better due to their O(1)access time.
* **For Frequent Insertions/Deletions:** If you often need to modify the list (insertions and deletions), linked lists are generally more efficient.