**Program – 2**

**AIM – Write an algorithm and program to implement Binary Search.**

**Algorithm –**

functionbinary\_search(A, n, T) is

L := 0

R := n − 1

while L ≤ R do

m := floor((L + R) / 2)

if A[m] < T then

L := m + 1

else if A[m] > T then

R := m − 1

else:

return m

return unsuccessful

**Source Code –**

#include <bits/stdc++.h>

using namespace std;

intbinarySearch(intarr[], int l, int r, int x)

{

if (r >= l) {

int mid = l + (r - l) / 2;

if (arr[mid] == x)

return mid;

if (arr[mid] > x)

returnbinarySearch(arr, l, mid - 1, x);

returnbinarySearch(arr, mid + 1, r, x);

}

return -1;

}

int main(void)

{

intarr[] = { 2, 3, 4, 10, 40 };

int x = 10;

int n = sizeof(arr) / sizeof(arr[0]);

int result = binarySearch(arr, 0, n - 1, x);

(result == -1) ? cout<< "Element is not present in array"

: cout<< "Element is present at index " << result;

return 0;

}

**2. Using Linked List**

**Source Code -**

#include<stdio.h>

#include<stdlib.h>

struct Node

{

int data;

struct Node\* next;

};

Node \*newNode(int x)

{

struct Node\* temp = new Node;

temp->data = x;

temp->next = NULL;

return temp;

}

struct Node\* middle(Node\* start, Node\* last)

{

if (start == NULL)

return NULL;

struct Node\* slow = start;

struct Node\* fast = start -> next;

while (fast != last)

{

fast = fast -> next;

if (fast != last)

{

slow = slow -> next;

fast = fast -> next;

}

}

return slow;

}

struct Node\* binarySearch(Node \*head, int value)

{

struct Node\* start = head;

struct Node\* last = NULL;

do

{

Node\* mid = middle(start, last);

if (mid == NULL)

return NULL;

if (mid -> data == value)

return mid;

else if (mid -> data < value)

start = mid -> next;

else

last = mid;

} while (last == NULL ||

last != start);

return NULL;

}

int main()

{

Node \*head = newNode(1);

head->next = newNode(4);

head->next->next = newNode(7);

head->next->next->next = newNode(8);

head->next->next->next->next = newNode(9);

head->next->next->next->next->next = newNode(10);

int value = 7;

if (binarySearch(head, value) == NULL)

printf("Value not present\n");

else

printf("Present");

return 0;

}