**Practical – 8**

AIM: Implementation and time analysis of Stack and its applications (infix, postfix, prefix).

Linear Search  
  
#include <iostream>

using namespace std;

int linearSearch(const int arr[], int n, int target)

{

for (int i = 0; i < n; ++i)

{

if (arr[i] == target)

{

return i;

}

}

return -1;

}

int main()

{

const int size = 8;

int array[size] = {4, 2, 8, 1, 7, 5, 3, 6};

int target = 5;

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

if (result != -1)

{

cout << "Linear Search: Target found at index " << result << endl;

}

else

{

cout << "Linear Search: Target not found in the array" << endl;

}

return 0;

}

**Binary Search**

#include<iostream>

using namespace std;

int binarySearch(const int arr[], int low, int high, int target) {

while (low <= high) {

int mid = low + (high - low) / 2;

if (arr[mid] == target) {

return mid;

} else if (arr[mid] < target) {

low = mid + 1;

} else {

high = mid - 1;

}

}

return -1;

}

int main() {

const int size = 8;

int array[size] = {11,22,1,3,4,9,99,44};

int target = 99;

int result = binarySearch(array, 0, size - 1, target);

if (result != -1) {

cout << "Binary Search: Target found at index " << result <<endl;

} else {

cout << "Binary Search: Target not found in the array" <<endl;

}

return 0;

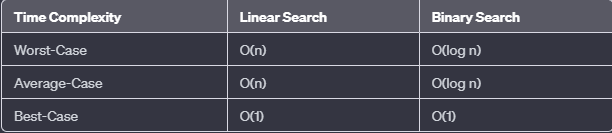
}

**OUTPUT**





Time analysis



Applications

**Linear Search:**

* Finding an element in an unsorted list.
* Checking for the presence of an item.
* Traversal of unordered data.

**Binary Search:**

* Searching in a sorted list or array.
* Finding the middle element.
* Dictionary-like searches.
* Efficient search in large databases.
* Spell checker in sorted word lists.
* Finding closest values.
* Implementation of data structures.
* Searching in virtual memory.