**PROGRAM 2**

**Implement Recursive Binary search and Linear search and determine the time required to search an element. Repeat the experiment for different values of N and plot a graph of the time taken versus N**

**Program:**

#include <stdio.h>

#include <stdlib.h>

#include <time.h>

int Linear\_Search(int array[10000], int search, int index, int n);

int Binary\_Search(int array[10000], int low,int high, int search);

int main()

{

clock\_t start,end;

double time;

int i,j,a,low, high, mid, n, search, array[10000],choice,loc,res,index;

printf("Enter the size of the array:\n");

scanf("%d", &n);

printf("The elements in the array are: \n");

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

{

array[i] = rand()%100;

printf("%d\t",array[i]);

}

printf("\n Enter the key to be searched: ");

scanf("%d",&search);

while(1){

printf("\nCHOICES\n");

printf("\n 1. Linear search");

printf("\n 2. Binary search");

printf("\n 3. Exit");

printf("\n Enter your choice: ");

scanf("%d",&choice);

switch(choice){

case 1:

start = clock();

loc = Linear\_Search(array, search, 0, n);

end = clock();

time = ((double)(end - start))/CLOCKS\_PER\_SEC;

printf("\nTime taken : %lf\n",time);

if (loc != 0)

{

printf("\nElement found in the array at location: %d\n", loc);

}

else

{

printf("\nElement not found in the array!");

}

break;

case 2 : for (i = 0; i < n; ++i)

{

for (j = i + 1; j < n; ++j)

{

if (array[i] > array[j])

{

a = array[i];

array[i] = array[j];

array[j] = a;

}

}

}

printf("\nsorted arrray: \n");

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

printf("%d\t", array[i]);

start = clock();

res = Binary\_Search(array, 0, n-1, search);

end = clock();

time = ((double)(end - start))/CLOCKS\_PER\_SEC;

printf("\nTime taken : %lf\n",time);

if(res == -1){

printf("Element is not found in array");

}

else{

printf("Element is present in the array at location: %d", res);

}

break;

case 3: exit(0);

break;

default: printf("/n Enter valid choice!");

break;

}

}

}

int Linear\_Search(int arr[], int search, int index, int n)

{

int pos = 0;

if(index >= n)

{

return 0;

}

else if (arr[index] == search)

{

pos = index + 1;

return pos;

}

else

{

return Linear\_Search(arr, search, index+1, n);

}

return pos;

}

int Binary\_Search(int arr[], int low, int high, int search)

{

if (high >= low)

{

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

if (arr[mid] == search) return mid;

if (arr[mid] > search) return Binary\_Search(arr, low, mid-1, search);

return Binary\_Search(arr, mid+1, high, search);

}

return -1;

}

**OUTPUT**

