**Assignment-1**

**EXPERIMENT – 1: Write a C program to implement the Binary Search and Linear Search**

**algorithm with array and linked list.**

**Binary Search and Linear Search algorithm with linked list.**

#include <stdio.h>

#include <stdlib.h>

void create();

void display();

int search(int key);

int Bsearch(struct node \*,int s,int r,int key);

struct node

{

int info,index;

struct node \*link;

} \*start,\*node1,\*tmp,\*node2;

int main()

{ int key,index,first,End,index1;

create();

printf("\n");

display();

printf("\nEnter element to search: ");

scanf("%d", &key);

index1=Bsearch(start,1,End,key);

index=search(key);

if (index >= 0)

printf("\nLinear Search:-\n%d found in the list at position %d\n", key, index + 1);

else

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

if (index1 >= 0)

printf("\nBinary Search:-\n%d found in the list at position %d\n", key, index + 1);

else

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

}

void create()

{

int n;

printf("Enter number of nodes to create:");

scanf("%d",&n);

printf("\nEnter the info to store in the node:-\n");

while(n>=1){

node1=(struct node \*)malloc(sizeof(start));

node1->link=NULL;

scanf("%d",&node1->info);

if(start==NULL)

{

start=node1;

}

else{

tmp=start;

while(tmp->link!=NULL){

tmp=tmp->link;

}

tmp->link=node1;

}

n--;

}

}

void display()

{

if (start == NULL)

{

printf("List is empty.\n");

return;

}

tmp = start;

while (tmp != NULL)

{

printf("%d ", tmp->info);

tmp = tmp->link;

}

printf("\n");

}

int search(int key)

{

int index;

index = 0;

node2 = start;

while (node2 != NULL && node2->info != key)

{

index++;

node2 = node2->link;

}

return (node2 != NULL) ? index : -1;

}

int Bsearch(struct node \*tmp,int s,int r,int key)

{

if(r-s==0||r-s==-1)

{

if(tmp->info==key)

{

return tmp->index ;

}

else if(tmp->link->info==key)

{

return tmp->link->index ;

}

else if(tmp->link->link->info==key)

{

return tmp->link->link->index;

}

}

else{

int mid;

if((s+(r-1))%2==0)

{

mid=(s+(r-1))/2;

}

else{

mid=((s+(r-1))-1)/2;

}

int end=mid;

while(mid!=0)

{

tmp=tmp->link;

mid--;

}

mid=end;

if(tmp->info<=key)

{

Bsearch(tmp,s,mid,key);

}

else{

tmp=start;

Bsearch(tmp,mid+1,r,key);

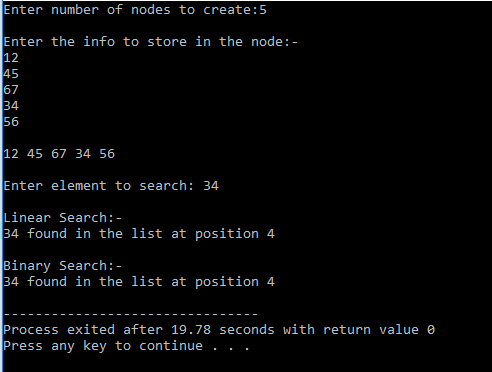
}

}

return 0;

}

**OUTPUT-**

****

**Binary Search and Linear Search algorithm with Array.**

#include <stdio.h>

#include <stdlib.h>

void linear\_search(int search\_key,int array[100],int n);

void binary\_search(int search\_key,int array[100],int n);

int main()

{

int array[100],search\_key,i,j,n,low,high,location;

printf("ENTER THE SIZE OF THE ARRAY:");

scanf("%d",&n);

printf("\nENTER THE ELEMENTS OF THE ARRAY:\n");

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

{

scanf("%d",&array[i]);

}

printf("\nENTER THE SEARCH KEY:");

scanf("%d",&search\_key);

linear\_search(search\_key,array,n);

binary\_search(search\_key,array,n);

return 0;

}

void linear\_search(int search\_key,int array[100],int n)

{

int i,location;

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

{

if(search\_key == array[i])

{

location = i;

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

printf("\nLinear Search:-\nThe location of Search Key = %d is %d\n",search\_key,location);

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

}

}

}

void binary\_search(int search\_key,int array[100],int n)

{

int mid,i,low,high;

low = 0;

high = n-1;

mid = (low + high)/2;

i=0;

while(search\_key != array[mid])

{

if(search\_key <= array[mid])

{

low = 1;

high = mid+1;

mid = (low+high)/2;

}

else

{

low = mid+1;

high = n;

mid = (low+high)/2;

}

}

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\nBinary Search:-\n");

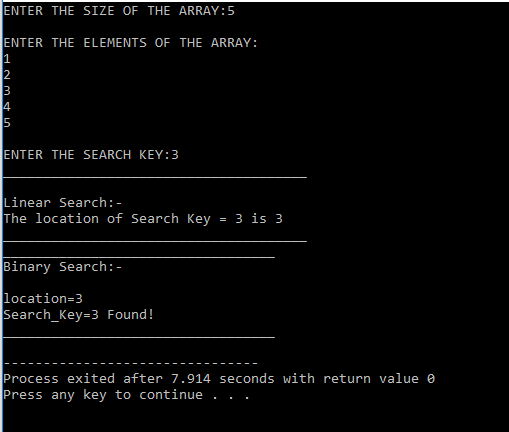
printf("\nlocation=%d\t",mid);

printf("\nSearch\_Key=%d Found!\n",search\_key);

printf("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\n");

}

**OUTPUT-**

****

**EXPERIMENT –2: Write a program to sort values in Ascending / Increasing order using Bubble Sort technique in linear array using recursion and without recursion.**

#include<stdio.h>

void BubbleSortRecursion(int a[],int num);

void bubbleSort(int a[], int num);

void BubbleSortRecursiondesc(int a[],int num);

main()

{

int i,j,num,temp;

printf("Enter number of elements:");

scanf("%d",&num);

int a[num];

printf("\nEnter numbers:-\n");

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

{

scanf("%d",&a[i]);

}

BubbleSortRecursion(a,num);

printf("\nSorting in Ascending Order with recursion:\n");

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

printf("%d ",a[i]);

printf("\n");

BubbleSortRecursiondesc(a,num);

printf("\nSorting in Descending Order with recursion:\n");

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

printf("%d ",a[i]);

printf("\n");

bubbleSort(a,num);

}

void BubbleSortRecursion(int a[],int num)

{

int i,j,temp;

if(num==1)

return;

for(j=0;j<num-1;j++)

if(a[j]>a[j+1])

{

temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

BubbleSortRecursion(a,num-1);

}

void BubbleSortRecursiondesc(int a[],int num)

{

int i,j,temp;

if(num==1)

return;

for(j=0;j<num-1;j++)

if(a[j]<a[j+1])

{

temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

BubbleSortRecursiondesc(a,num-1);

}

void bubbleSort(int a[], int num)

{ int i,temp,j;

for(i=0; i<(num-1); i++)

{

for(j=0; j<(num-i-1); j++)

{

if(a[j]>a[j+1])

{

temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

}

printf("\nSorting in Ascending Order without recursion:\n");

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

printf("%d ",a[i]);

printf("\n");

for(i=0; i<(num-1); i++)

{

for(j=0; j<(num-i-1); j++)

{

if(a[j]<a[j+1])

{

temp=a[j];

a[j]=a[j+1];

a[j+1]=temp;

}

}

}

printf("\nSorting in Descending Order without recursion:\n");

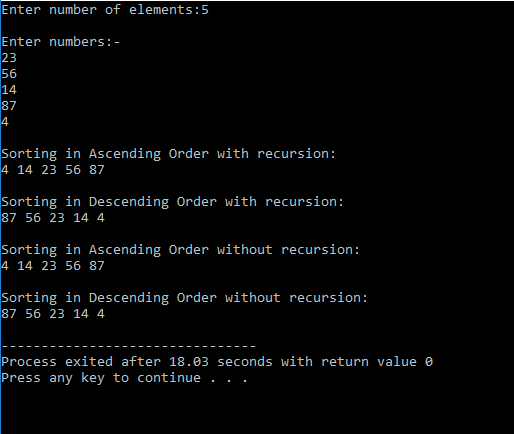
for(i=0; i<num; i++)

printf("%d ",a[i]);

printf("\n");

}

**OUTPUT-**

****