UIET MDU

Rohtak

DS Lab File

Submitted To:

Ms. Amita Dhankhar

Submitted By:

Payal

Roll No.23609

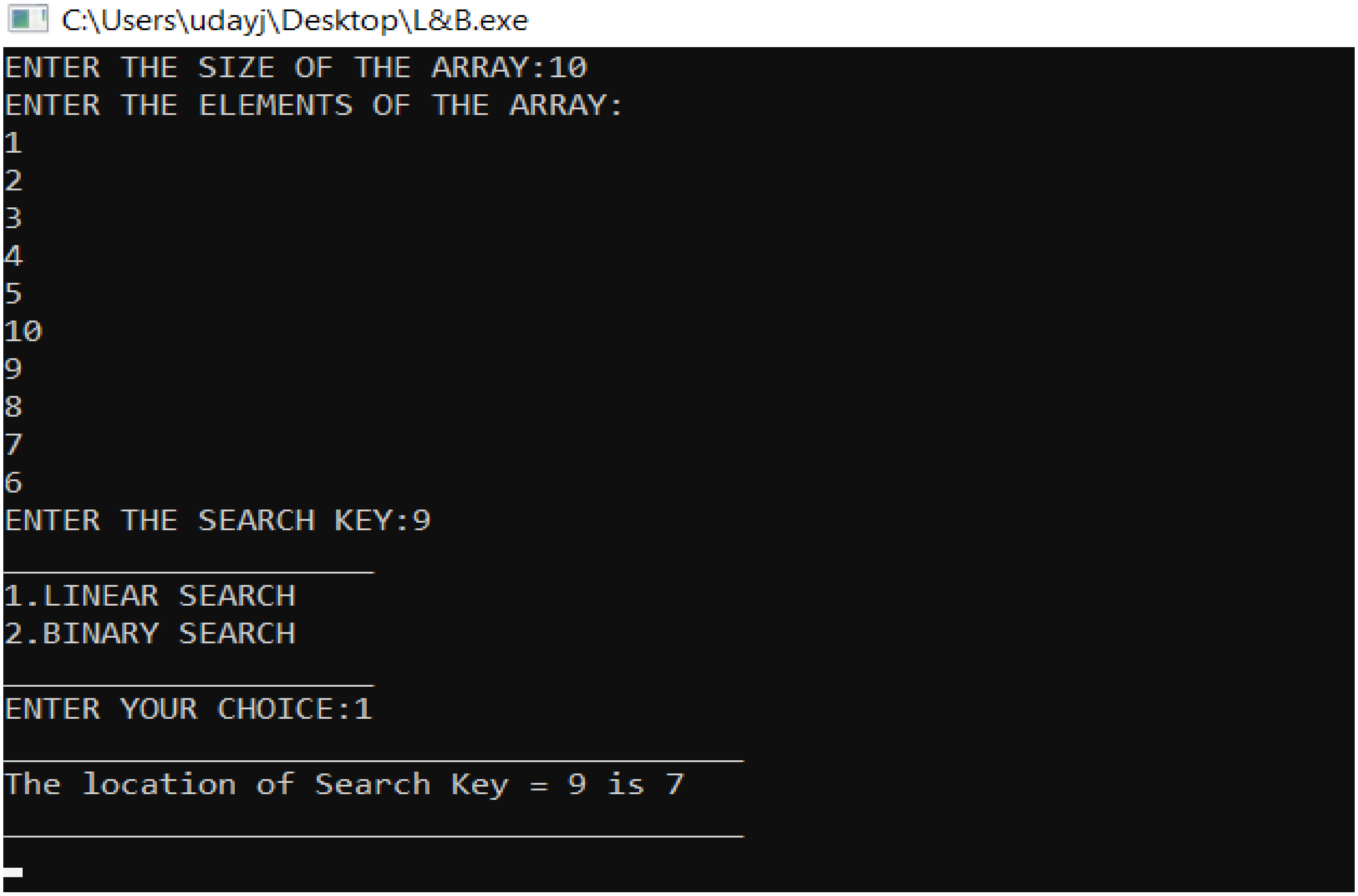
CSE-II (3rd Sem.)

/* To Search An Element Using Linear & Binary Search */

```
#include <stdio.h>

#include <stdlib.h>

main()
{

/* Declare variables - array_of_number,search_key,i,j,low,high*/

   int array[100],search_key,i,j,n,low,high,location,choice;

   void linear_search(int search_key,int array[100],int n);

   void binary_search(int search_key,int array[100],int n);


/* read the elements of array */

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

   scanf("%d",&n);

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

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

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

   }

/* Get the Search Key element for Linear Search */

   printf("ENTER THE SEARCH KEY:");

   scanf("%d",&search_key);

/* Choice of Search Algorithm */

   printf("___________________\n");

   printf("1.LINEAR SEARCH\n");

   printf("2.BINARY SEARCH\n");
```

```
    printf("___________________\n");

    printf("ENTER YOUR CHOICE:");

    scanf("%d",&choice);

    switch(choice)
    {

    case 1:

        linear_search(search_key,array,n);

        break;

    case 2:

        binary_search(search_key,array,n);

        break;

    default:

        exit(0);

}

    getch();

    return 0;

}

/* LINEAR SEARCH */

    void linear_search(int search_key,int array[100],int n)
    {

/*Declare Variable */

        int i,location;

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

            if(search_key == array[i])
            {
                location = i;

    printf("________________________________________\n");

    printf("The location of Search Key = %d is %d\n",search_key,location);

    printf("________________________________________\n");

        }

    }

}

/* Binary Search to find Search Key */

void binary_search(int search_key,int array[100],int n)
{

    int mid,i,low,high;

    low = 1;

    high = n;

    mid = (low + high)/2;

    i=1;

    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("_______________________________________\n");

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

      printf("Search_Key=%d Found!\n",search_key);

      printf("_______________________________________\n");

    }
```

OUTPUT:-



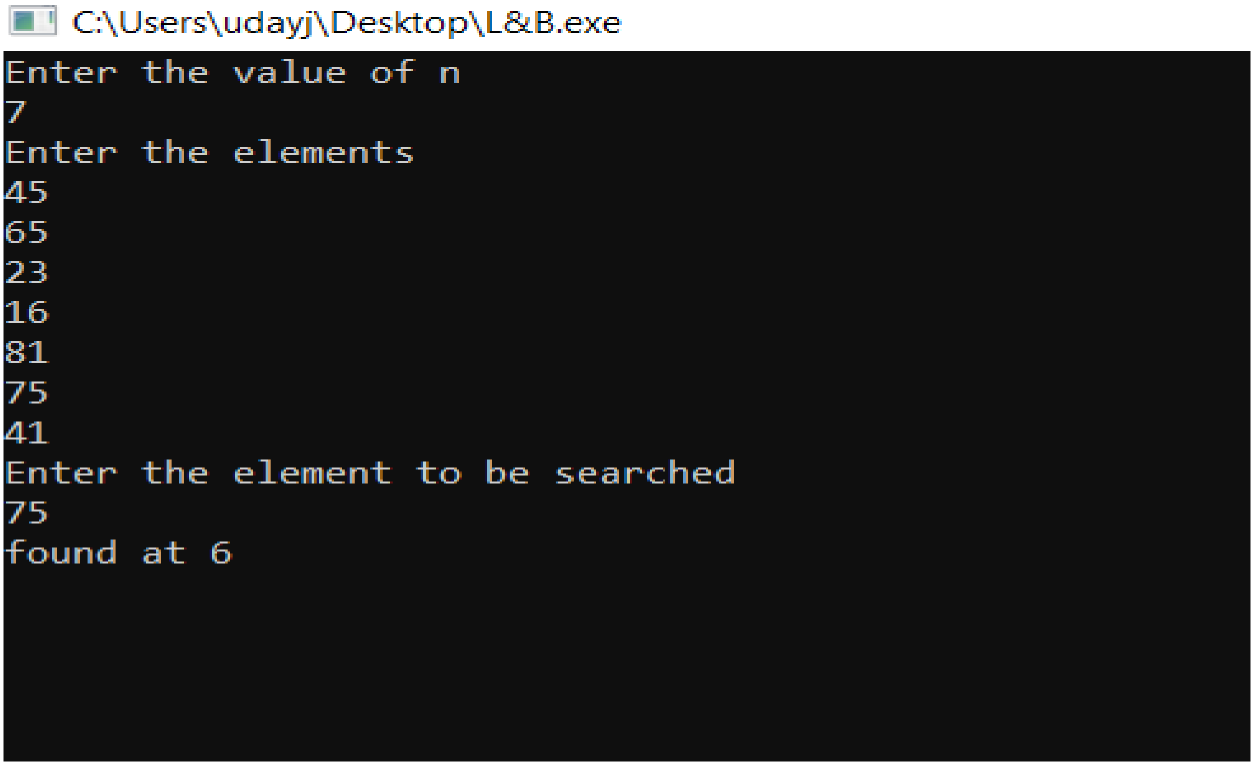
C:\Users\udayj\Desktop\L&B.exe

```
ENTER THE SIZE OF THE ARRAY:10
ENTER THE ELEMENTS OF THE ARRAY:
1
2
3
4
5
10
9
8
7
6
ENTER THE SEARCH KEY:9
___________________
1.LINEAR SEARCH
2.BINARY SEARCH
___________________
ENTER YOUR CHOICE:1
______________________________________
The location of Search Key = 9 is 7
______________________________________
```

/* To Search An Element Using Binary Search - Recursive Method */

```
#include<stdio.h>
#include<conio.h>
int bin(int b[],int low,int high,int item);
void main()
{
int a[10],i,mid,c,lb,ub,n,h;
printf("Enter the value of n \n");
scanf("%d",&n);
printf("Enter the elements\n");
for(i=1;i<=n;i++)
scanf("%d",&a[i]);
printf("Enter the element to be searched\n");
scanf("%d",&h);
lb=1,ub=n;
c=bin(a,lb,ub,h);
if(c==0)
{
printf("not found");
}
else
printf("found at %d",c);
getch();
}
int bin(int b[10],int low,int high,int item)
{
int mid,loc=0;
mid= (low +high)/2;
if(low>high)
{
return loc;
}
if(b[mid]==item)
{
loc=mid;
}
else if(b[mid]>item)
{
high = mid-1;
loc=bin(b,low,high,item);
}
else if(b[mid]<item)
{
low=mid +1;
loc=bin(b,low,high,item);
}
else
{
loc=0;
}
return loc;
}
```

OUTPUT:-

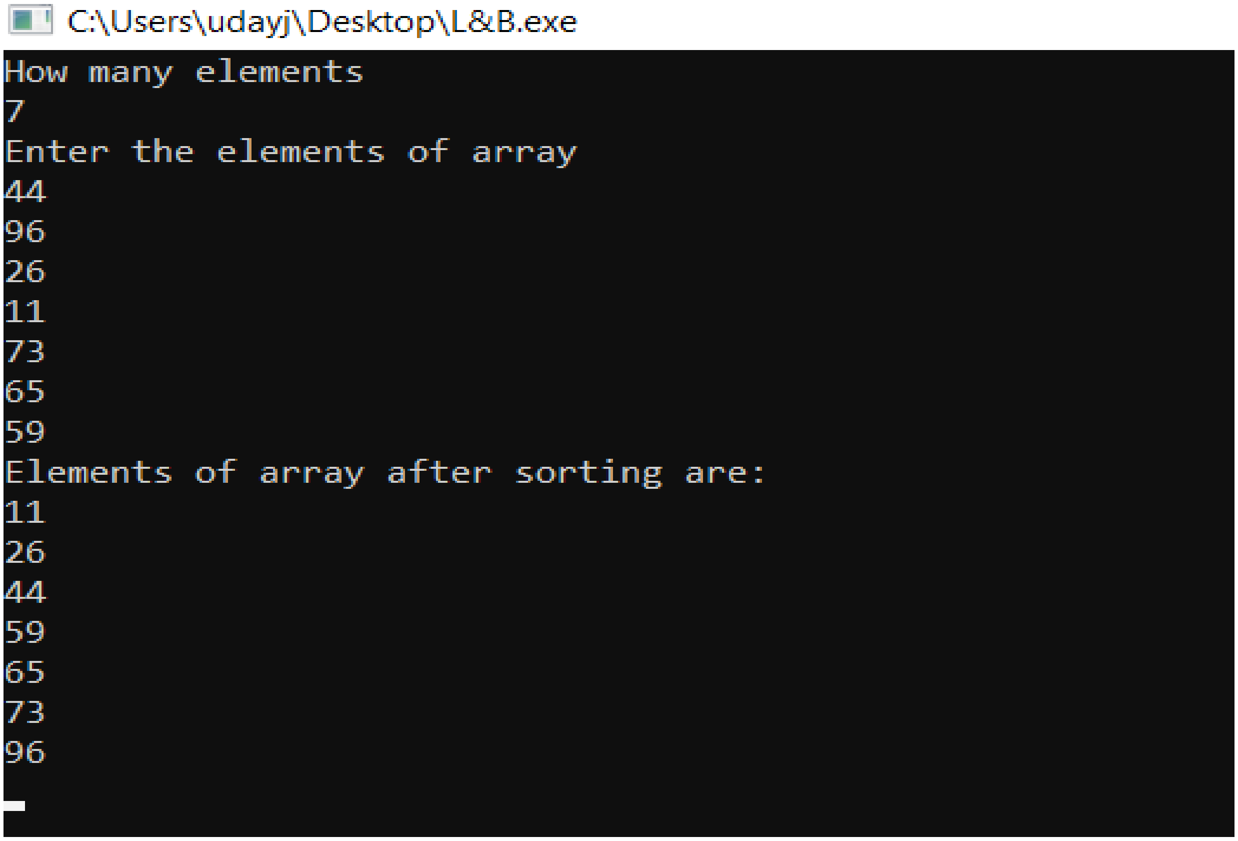


```
Enter the value of n
7
Enter the elements
45
65
23
16
81
75
41
Enter the element to be searched
75
found at 6
```

/* To Sort Elements Using Insertion Sort */

```
#include<stdio.h>
#include<conio.h>
void main()
{
        int a[100],n,k,i,j,temp;
        clrscr();
        printf("How many elements");
        scanf("%d",&n);
        printf("Enter the elements of array");
        for(i=0;i<= n-1; i++)
        {
                scanf("%d",&a[i]);
        }
        for(k=1; k<= n-1 ;k++)
        {
                temp = a[k];
                j= k-1;
                while((temp<a[j]) &&(j>=0))
                {
                        a[j+1] = a[j];
                        j = j-1;
                }
                a[j+1]= temp;
        }



        printf("Elements of array after sorting are:\n");
        for(i=0; i<n ; i++)
        {
                printf("%d\n" ,a[i]);
        }
        getch();
}
```

OUTPUT:-

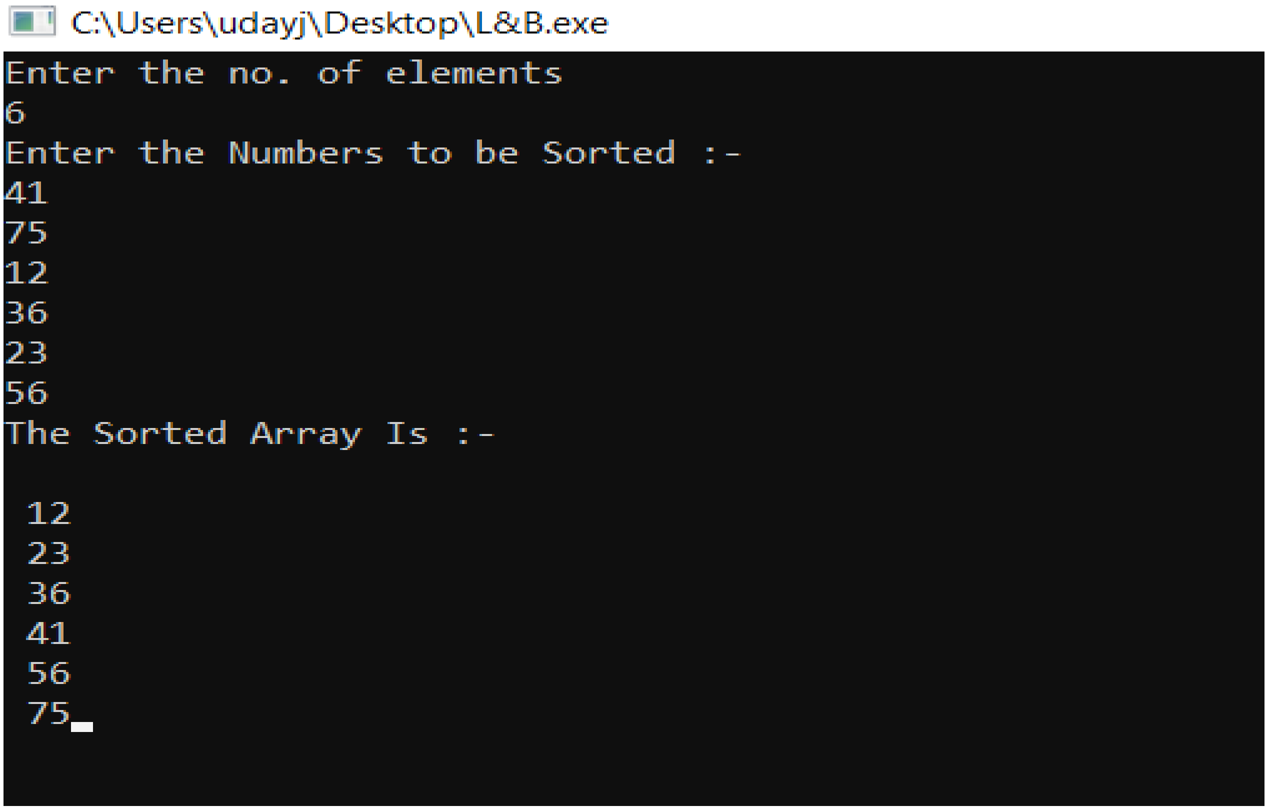


```
How many elements
7
Enter the elements of array
44
96
26
11
73
65
59
Elements of array after sorting are:
11
26
44
59
65
73
96

```

/* To Sort Elements Using Bubble Sort */

```
#include<stdio.h>
#include<conio.h>
void main()
{
int a[10],i,j,temp=0,n;
printf("Enter the no. of elements \n");
scanf("%d",&n);
printf("Enter the Numbers to be Sorted :- \n");
for(i=0;i<n;i++)
{
scanf("%d",&a[i]);
}
printf("The Sorted Array Is :- \n");
for(i=0;i<n;i++)
{
for(j=0;j<n-i;j++)
{
if(a[j]>a[j+1])
{
temp=a[j+1];
a[j+1]=a[j];
a[j]=temp;
}
}
}
for(i=0;i<n;i++)
printf("\n %d",a[i]);
getch();
}
```

OUTPUT :-



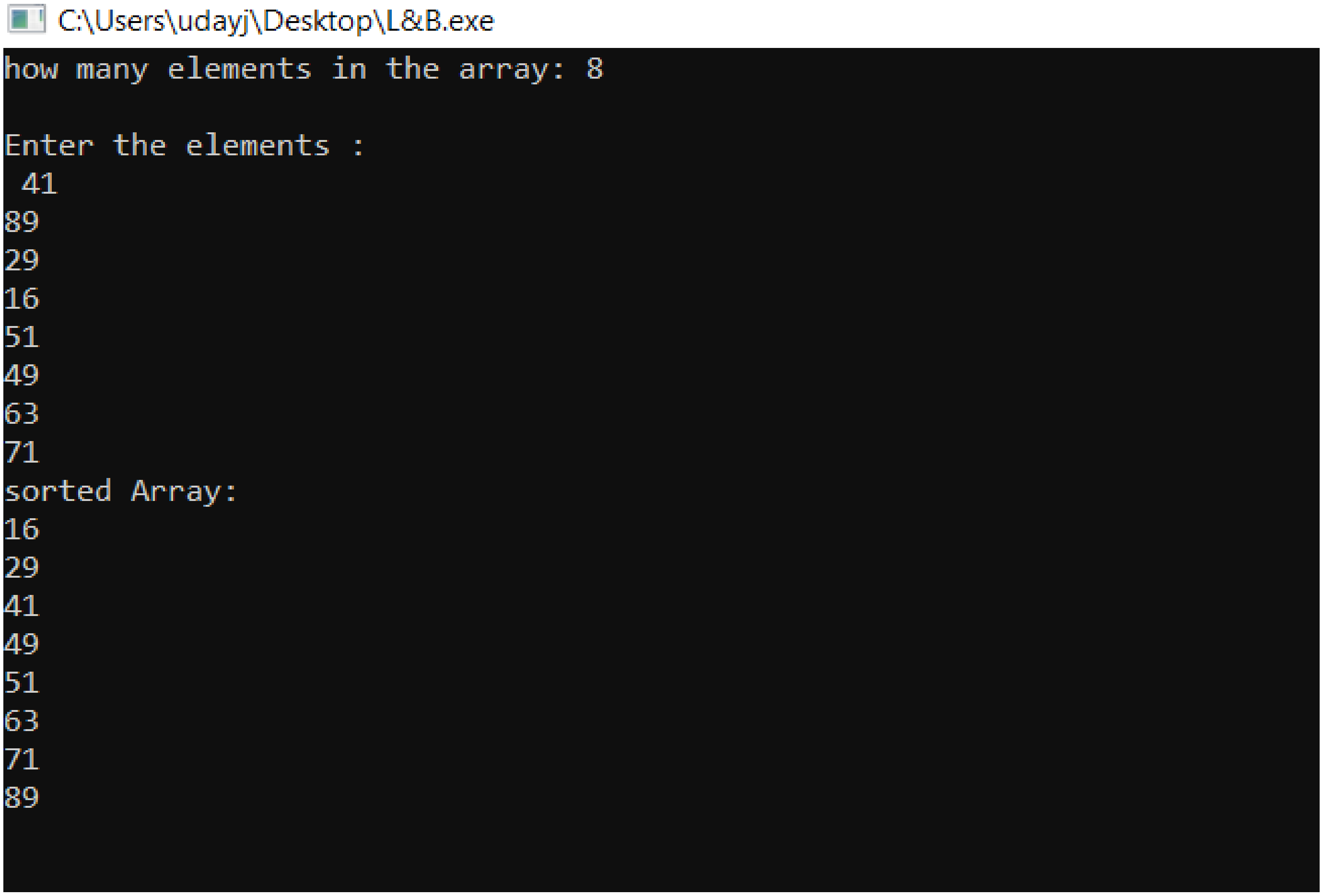
```
Enter the no. of elements
6
Enter the Numbers to be Sorted :-
41
75
12
36
23
56
The Sorted Array Is :-

 12
 23
 36
 41
 56
 75_
```

/* To Sort Elements Using Quick Sort */

```
#include<stdio.h>
#include<conio.h>
#define max 100
int a[max],n,i,l,h;
void main()
{
void input (void);

        input();
        getch();
}
void input(void)
{
void quick_sort (int a[], int l ,int h);
void output ( int a[], int n);
printf ("how many elements in the array: ");
scanf("%d" ,&n);
printf("\n");
printf("Enter the elements : \n ");
for (i =0; i <= n-1; i++)
{
scanf("%d", &a[i]);
}
l = 0;
h =n-1;
quick_sort (a,l,h);
printf("sorted Array: \n");
output (a,n);
}
void quick_sort(int a[], int l, int h)
{
 int temp,key, low, high;
 low=l;
 high=h;
 key = a[(low+high)/2];
 do
 {
 while(key >a[low])
 {
 low++;
 }
 while (key < a[high])
 {
 high--;
 }
 if(low<= high)
 {
 temp=a[low];
 a[low++] =a[high];
 a[high--] =temp;
 }
        }
 while(low<=high);
 if(l<high)
 {
 quick_sort(a,l,high);
 }
 if (low<h)
 {
 quick_sort(a,low,h);
 }
 }
 void output (int a[], int n)
```

```
{
for (i=0 ; i<= n-1 ; i++)
{
printf("%d\n",a[i]);
}
}
```

OUTPUT:-



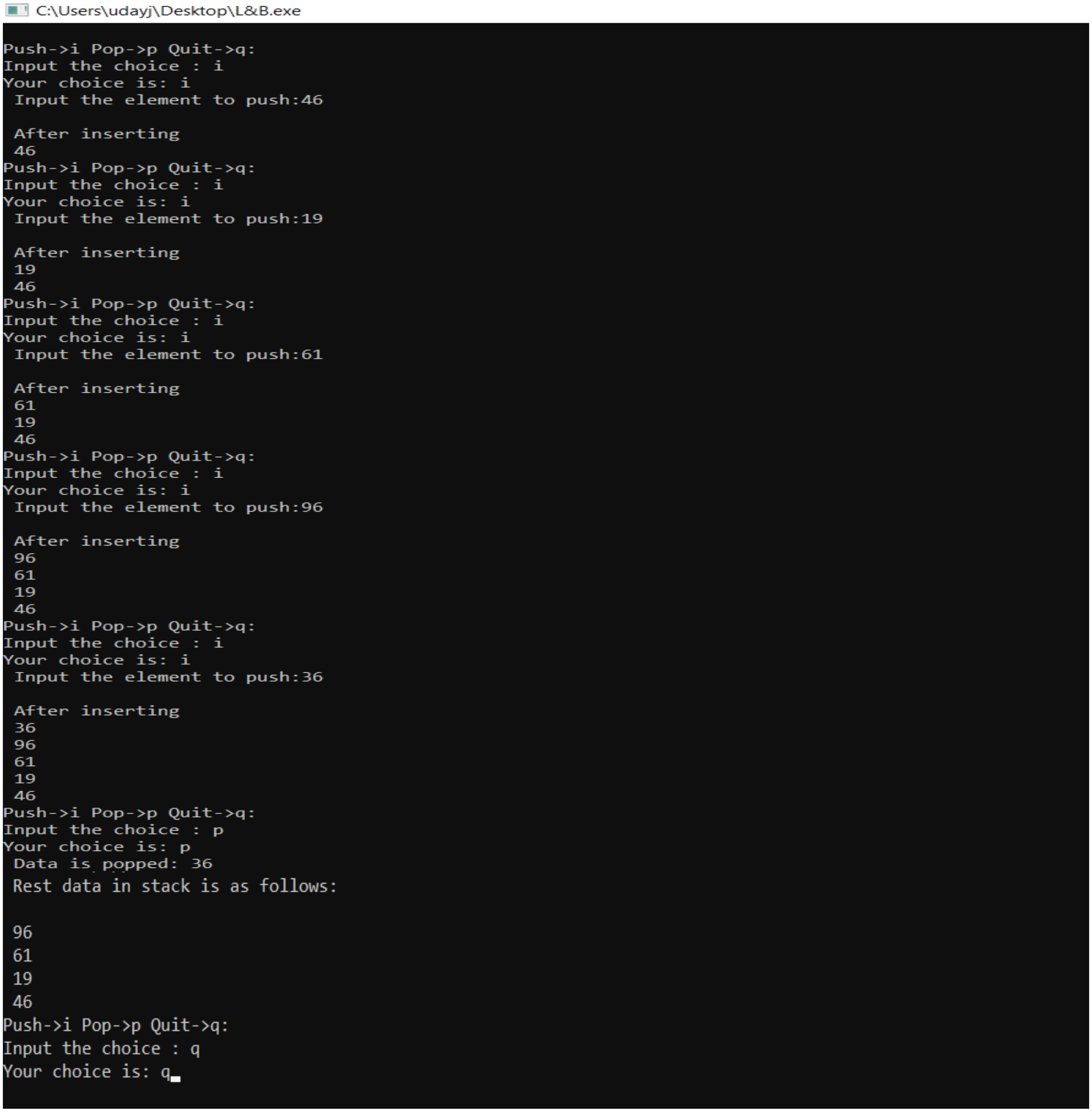
```
how many elements in the array: 8

Enter the elements :
 41
89
29
16
51
49
63
71
sorted Array:
16
29
41
49
51
63
71
89
```

/* To Perform Push, Pop & Display Operations On Stack Using Array */

```
# include<stdio.h>
# include<string.h>
# include<ctype.h>
# define size 100
int top = -1;
int flag = 0;
int stack[size];
void push(int *, int);
int pop(int *);
void display(int *);
/* Definition of the push function */
void push(int s[], int d)
{
	if(top ==(size-1))
		flag = 0;
	else
	{
		flag = 1;
		++top;
		s[top] = d;
	}
}



/* Definition of the pop function */
int pop(int s[])
{
	int popped_element;
	if(top == -1)
	{
```

```
                popped_element = 0;
                flag = 0;
        }
        else
        {
                flag = 1;
                popped_element = s[top];
                --top;
        }
        return (popped_element);
}
/* Definition of the display function */
void display(int s[])
{
        int i;
        if(top == -1)
        {
                printf("\n Stack is empty");
        }
        else
        {
                for(i = top; i >= 0; --i)
                        printf("\n %d", s[i] );
        }
}
/* Function main */
void main()
{
        int  data;
        char choice;
        int q = 0;
        int top = -1;
        do
        {
                printf(" \nPush->i Pop->p Quit->q:");
                printf("\nInput the choice : ");
                do
                {
                        choice = getchar();
                        choice =tolower(choice);
                }while(strchr("ipq",choice)==NULL);
                printf("Your choice is: %c",choice);
                switch(choice)
                {
                case 'i' :
                        printf("\n Input the element to push:");
                        scanf("%d", &data);
                        push(stack, data);
                        if(flag)
                        {
                            printf("\n After inserting ");
                            display(stack);
                            if(top == (size-1))
                                printf("\n Stack is full");
                        }
                        else
                            printf("\n Stack overflow after pushing");
                        break;
                case 'p' :
                        data = pop(stack);
                        if(flag)
                        {
                            printf("\n Data is popped: %d", data);
                            printf("\n Rest data in stack is as follows:\n");
                            display(stack);
                        }
                        else
                            printf("\n Stack underflow" );
                        break;
                case 'q':
                        q = 1;
                }
        } while(!q);
        getch();
}
```

OUTPUT:-

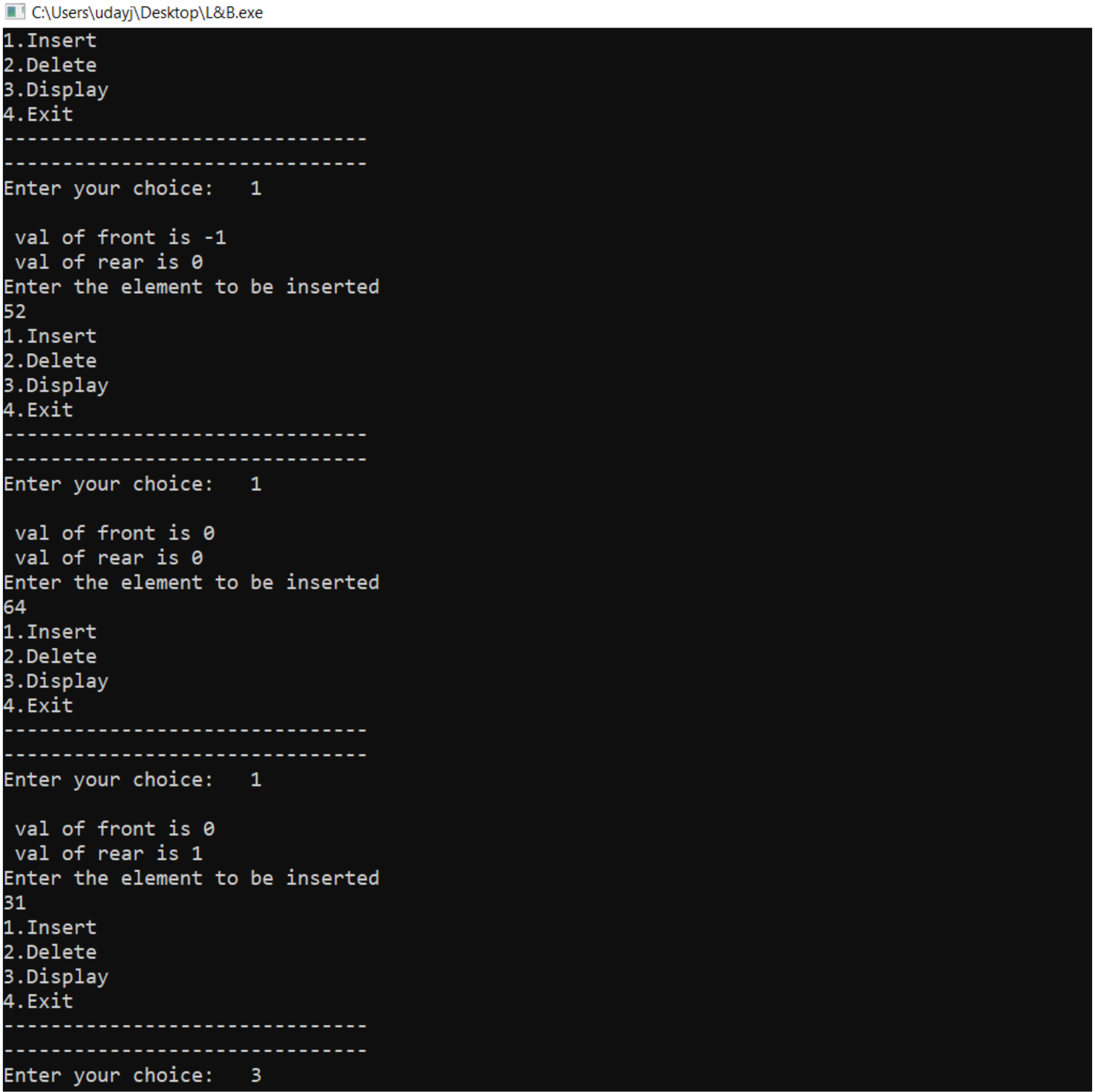


```
Push->i Pop->p Quit->q:
Input the choice : i
Your choice is: i
 Input the element to push:46

 After inserting
 46
Push->i Pop->p Quit->q:
Input the choice : i
Your choice is: i
 Input the element to push:19

 After inserting
 19
 46
Push->i Pop->p Quit->q:
Input the choice : i
Your choice is: i
 Input the element to push:61

 After inserting
 61
 19
 46
Push->i Pop->p Quit->q:
Input the choice : i
Your choice is: i
 Input the element to push:96

 After inserting
 96
 61
 19
 46
Push->i Pop->p Quit->q:
Input the choice : i
Your choice is: i
 Input the element to push:36

 After inserting
 36
 96
 61
 19
 46
Push->i Pop->p Quit->q:
Input the choice : p
Your choice is: p
 Data is popped: 36
 Rest data in stack is as follows:

 96
 61
 19
 46
Push->i Pop->p Quit->q:
Input the choice : q
Your choice is: q
```

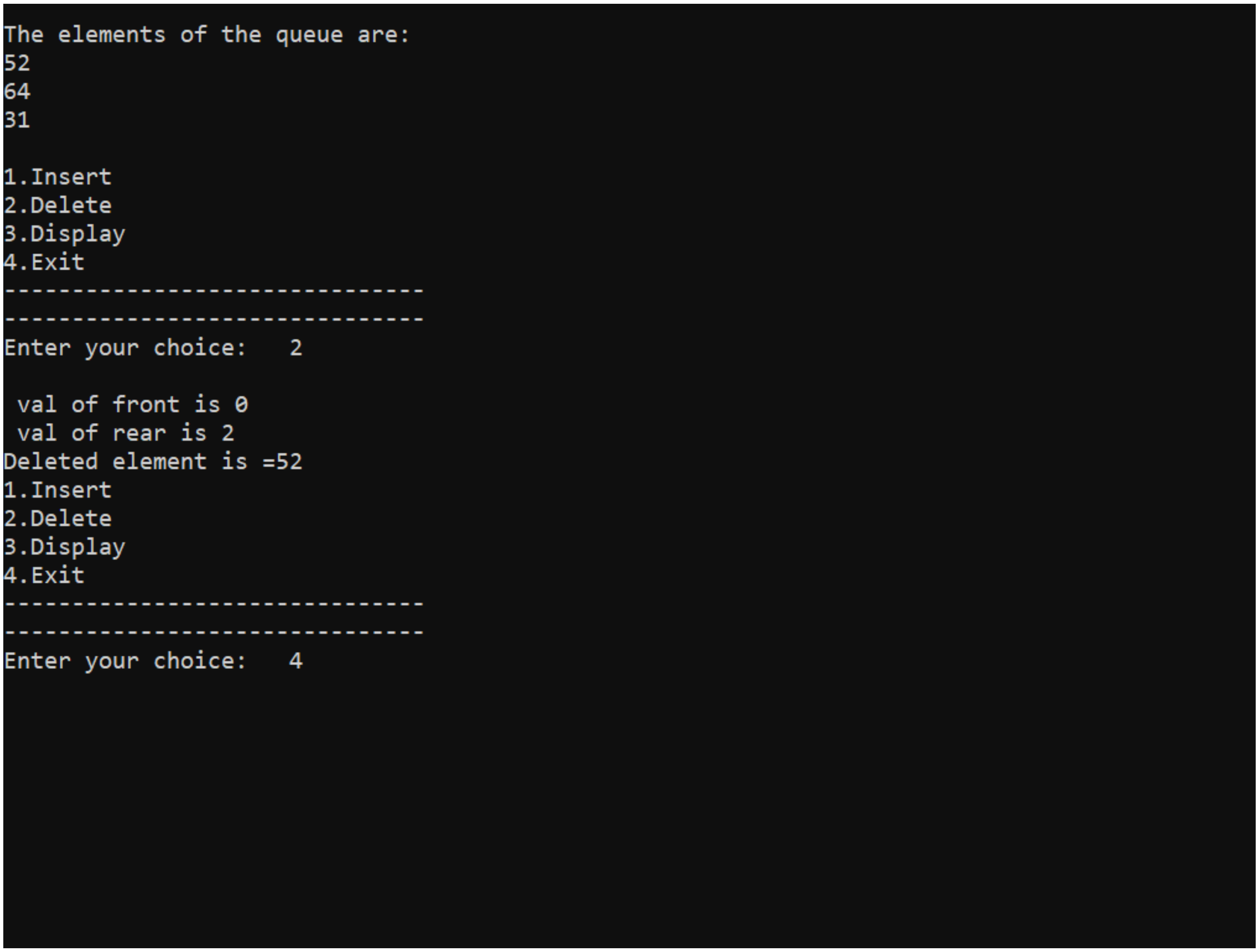
/* To perform insertion , deletion & display operations on circular queue using array */

```
#include<stdio.h>
#include<conio.h>
#define MAXSIZE 5
int cq[10];
int front=-1,rear=0;
int choice;
char ch;
void main()
{
        clrscr();
        do
        {
                   printf("1.Insert\n");
                   printf("2.Delete\n");
                   printf("3.Display\n");
                   printf("4.Exit\n");
                   printf("------------------------------");
                   printf("\n------------------------------");
                   printf("\nEnter your choice:   ");
                   scanf("%d",&choice);
                    switch(choice)
                   {
                        case 1 : cqinsert();
                                break;
                        case 2 : cqdelete();
                                break;
                        case 3 : cqdisplay();
                                break;
                        case 4: exit();
                }
        }//end of do
          while(choice!=4);
 }// end of main()
cqinsert()
{
     int num;
     printf("\n val of front is %d",front);
     printf("\n val of rear is %d",rear);
     if(front==(rear+1)%MAXSIZE)
         {
                 printf("\nQueue is full\n");
                 return;
         }
    else
          {
                    printf("\nEnter the element to be inserted\n");
                    scanf("%d",&num);
                    if(front==-1)
                         front=rear=0;
                   else
                            rear=(rear+1) % MAXSIZE;
                            cq[rear]= num;
          }
 return;
}
int cqdelete()
{
     int num;
     printf("\n val of front is %d",front);
     printf("\n val of rear is %d",rear);
     if(front==-1)
         {
                  printf("\n\tQueue is Empty\n");
                  return;
          }
     else
        {
                 num=cq[front];
                 printf("\nDeleted element is =%d\n",cq[front]);
                 if(front==rear) //when there is one element in the queue
                        front=rear=-1;
                 else
                        front=(front+1)%MAXSIZE;
        }
         return(num);
}
 cqdisplay()
{
        int i;
        if(front==-1)
        {
                printf("Queue is empty\n");
                return;
        }
        else
        {
                printf("\nThe elements of the queue are:   \n");
                for(i=front;i<=rear;i++)
                {
                        printf("%d\n",cq[i]);
                }
        }
        if(front>rear)
        {
                for(i=front;i<MAXSIZE;i++)
                {
                        printf("%d\n",cq[i]);
                }
                for(i=0;i<=rear;i++)
                {
                        printf("%d\n",cq[i]);
                }
        }
        printf("\n");
}
```

OUTPUT :-



```
1.Insert
2.Delete
3.Display
4.Exit
-------------------------------
-------------------------------
Enter your choice:   1

 val of front is -1
 val of rear is 0
Enter the element to be inserted
52
1.Insert
2.Delete
3.Display
4.Exit
-------------------------------
-------------------------------
Enter your choice:   1

 val of front is 0
 val of rear is 0
Enter the element to be inserted
64
1.Insert
2.Delete
3.Display
4.Exit
-------------------------------
-------------------------------
Enter your choice:   1

 val of front is 0
 val of rear is 1
Enter the element to be inserted
31
1.Insert
2.Delete
3.Display
4.Exit
-------------------------------
-------------------------------
Enter your choice:   3
```



```
The elements of the queue are:
52
64
31

1.Insert
2.Delete
3.Display
4.Exit
-------------------------------
-------------------------------
Enter your choice:   2

 val of front is 0
 val of rear is 2
Deleted element is =52
1.Insert
2.Delete
3.Display
4.Exit
-------------------------------
-------------------------------
Enter your choice:   4
```

/* To perform insertion , deletion & display operations on linear linked list */

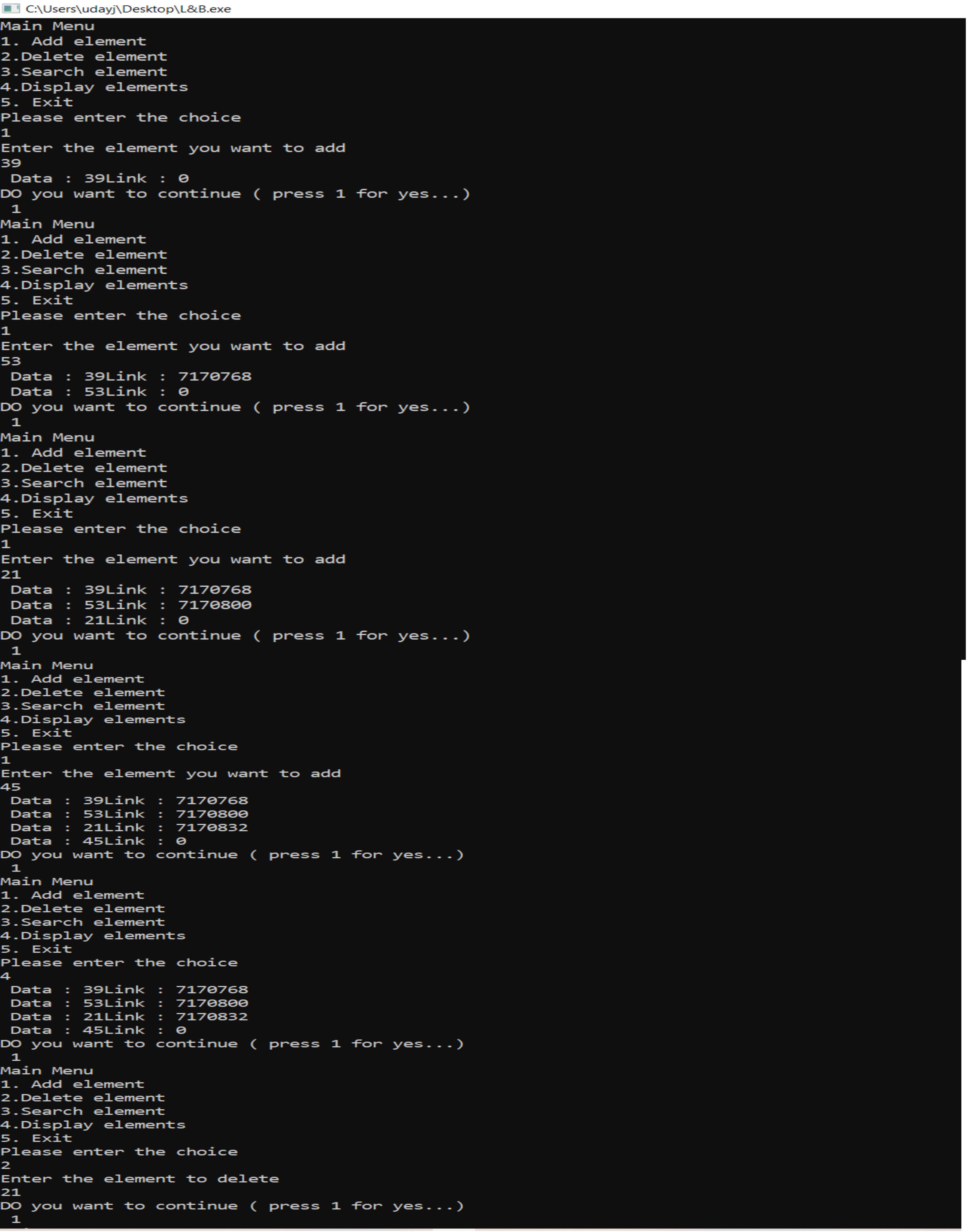
```
# include<stdio.h>
# include<conio.h>
# include "malloc.h"
struct node
{
int data;
struct node *link;
};

void main()
{
int will,wish,num;
struct node *ptr,*ptr2,*result,*temp;
void add(struct node **,int );
struct node * search(struct node *);
void display(struct node *);
void del(struct node *,int);
ptr='\0';
ptr2='\0';
result='\0';
will=1;
while(will==1)
{
printf("Main Menu \n");
printf("1. Add element \n");
printf("2.Delete element \n");
printf("3.Search element \n");
printf("4.Display elements \n");
printf("5. Exit \n");
printf("Please enter the choice \n");
scanf("%d",&wish);
switch(wish)
{
case 1:
                    printf("Enter the element you want to add \n");
                    scanf("%d",&num);
                    add(&ptr,num);
                    display(ptr);
```

```
                        break;
case 2:
                        printf("Enter the element to delete \n");
                        scanf("%d",&num);
                        del(ptr,num);
                        break;
case 3:
                        printf("Now demonstrating search \n");
                        temp = search(ptr);
                        printf("Address of first occurence is  %u ",temp);
                        break;



case 4:
                        display(ptr);
                        break;
                        case 5:
                        exit(1);
default:
                        printf("Illegal choice \n");
}
printf("DO you want to continue ( press 1 for yes...)\n ");
scanf("%d",&will);
}
}

// adding data in the linked list//
void add(struct node **q,int num)
{
struct node *temp;
temp = *q;
if(*q=='\0')
{
            *q=malloc(sizeof(struct node));
            temp = *q;
}
else
{
            while((temp->link)!='\0')
            {
                  temp=temp->link;
```

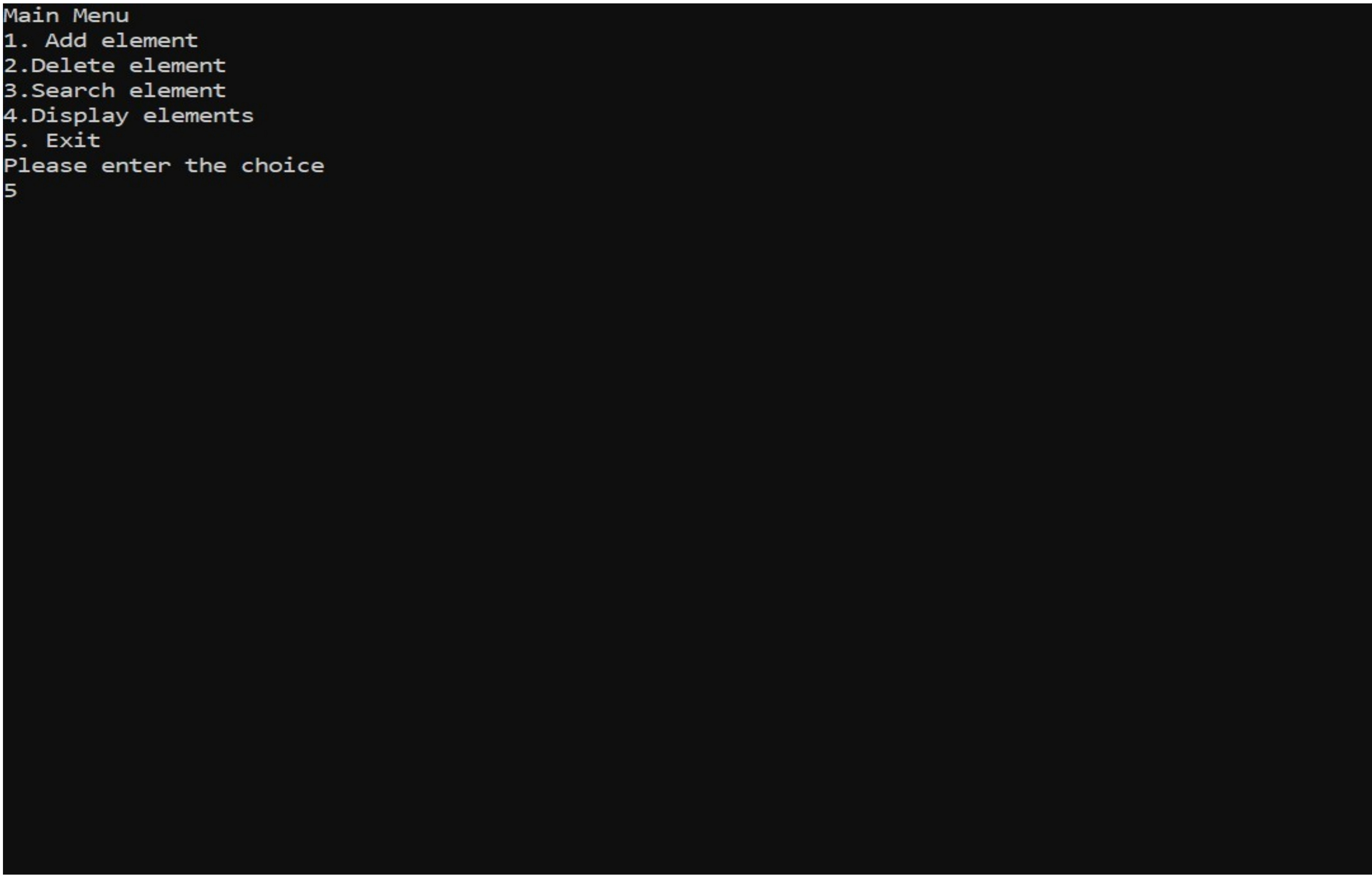
```
        }
        temp->link = malloc(sizeof(struct node));
        temp=temp->link;
}
temp->data = num;
temp->link  = '\0';
}

// display data from the linked list//
void display(struct node *pt)
{
while(pt!='\0')
{
printf(" Data : %d",pt->data);
        printf("Link : %d",pt->link);
        printf("\n");
        pt=pt->link;
}
}

/* searching an element in the linked list and this function finds the first
occurence of
  of the data and returns a pointer to its address*/
struct node * search(struct node *p)
{
struct node *temp;
int num;
temp = p;
printf("Enter the data that you want to search \n");
scanf("%d",&num);
printf("Link of temp %u", temp->link);
while(temp->link!='\0')
    {
        printf(" In while \n");
        if(temp->data == num)
        return(temp);
        temp=temp->link;
    }
return('\0');
}
```

```
// deleting data from the linked list//
void del(struct node *p,int num)
{
struct node *temp,*x;
temp=p;
x= '\0';
while (temp->link !='\0')
{
if(temp->data == num)
{
                        if (x=='\0')
                        {
                                p = temp->link;
                                free(temp);
                                return;
                        }
                        else
                        {
                                x->link = temp->link;
                                free(temp);
                                return;
                        }
}
x=temp;
temp=temp->link;
}
printf("No such entry to delete \n");
}
```

OUTPUT:-



```
Main Menu
1. Add element
2.Delete element
3.Search element
4.Display elements
5. Exit
Please enter the choice
1
Enter the element you want to add
39
 Data : 39Link : 0
DO you want to continue ( press 1 for yes...)
 1
Main Menu
1. Add element
2.Delete element
3.Search element
4.Display elements
5. Exit
Please enter the choice
1
Enter the element you want to add
53
 Data : 39Link : 7170768
 Data : 53Link : 0
DO you want to continue ( press 1 for yes...)
 1
Main Menu
1. Add element
2.Delete element
3.Search element
4.Display elements
5. Exit
Please enter the choice
1
Enter the element you want to add
21
 Data : 39Link : 7170768
 Data : 53Link : 7170800
 Data : 21Link : 0
DO you want to continue ( press 1 for yes...)
 1
Main Menu
1. Add element
2.Delete element
3.Search element
4.Display elements
5. Exit
Please enter the choice
1
Enter the element you want to add
45
 Data : 39Link : 7170768
 Data : 53Link : 7170800
 Data : 21Link : 7170832
 Data : 45Link : 0
DO you want to continue ( press 1 for yes...)
 1
Main Menu
1. Add element
2.Delete element
3.Search element
4.Display elements
5. Exit
Please enter the choice
4
 Data : 39Link : 7170768
 Data : 53Link : 7170800
 Data : 21Link : 7170832
 Data : 45Link : 0
DO you want to continue ( press 1 for yes...)
 1
Main Menu
1. Add element
2.Delete element
3.Search element
4.Display elements
5. Exit
Please enter the choice
2
Enter the element to delete
21
DO you want to continue ( press 1 for yes...)
 1
```



```
Main Menu
1. Add element
2.Delete element
3.Search element
4.Display elements
5. Exit
Please enter the choice
5
```

/*To perform insertion , deletion , searching & traversal operations in Binary Search Tree */

```
#include<stdio.h>
#include<stdlib.h>

struct btnode
{
        int value;
        struct btnode *l;
        struct btnode *r;
} *root = NULL , *temp = NULL, *t2, *t1;

void delete1();
void insert();
void delete();
void inorder(struct btnode *t);
void create();
void search(struct btnode *t);
void preorder(struct btnode *t);
void postorder(struct btnode *t);
void search1(struct btnode *t,int data);
int smallest(struct btnode *t);
int largest(struct btnode *t_);

int flag = 1;
void main()
{
        int ch;
        printf("\nOPERATIONS--- ");
        printf("\n1 - Insert an element into tree\n");
        printf("\n2 - Delete an element from the tree\n");
        printf("\n3 - Inorder Traversal\n");
        printf("\n4 - Preorder Traversal\n");
        printf("\n5 - Postorder Traversal\n");
        printf("\n6 - Exit\n");
        while(1)
        {
                printf("\nEnter your choice : ");
                scanf("%d", ch);
                switch(ch)
{
            case 1:
                insert();
                break;
            case 2:
                delete();
                break;
            case 3:
                inorder(root);
                break;
            case 4:
                preorder(root);
                break;
            case 5:
                postorder(root);
                break;
            case 6:
                exit(0);
            default:
                printf("Wrong choice, Please enter correct choice ");
                break;

        }
    }
}

void insert()
{
    create();
    if(root == NULL)
        root = temp;
    else
        search(root);
}

void create()
{
```

```
int data;
        printf("Enter data of node to be inserted : ");
        scanf("%d",&data);
        temp = (struct btnode *)malloc(1*sizeof(struct btnode));
        temp->l = temp->r = NULL;
}

void inorder(struct btnode *t)
{
        if(root == NULL)
        {
                printf("No elements in a tree to display");
                return;
        }
        if(t->r != NULL)
        {
                inorder(t->r);
        }
        printf("%d -> ", t->value);
        if(t->r != NULL)
        {
                inorder(t->r);
        }
}



void search(struct btnode *t)
{
        if((temp->value > t->value)&&(t->r != NULL))
              search(t->r);
        else if((temp->value > t->value)&&(t->r == NULL))
               t->r = temp;
        else if((temp->value < t->value)&&(t->l != NULL))
               search(t->l);
        else if((temp->value < t->value)&&(t->l == NULL))
```

```
t->l = temp;
}



void delete()
{
        int data;
        if(root == NULL)
        {
                printf("No elements in a tree to delete:");
                return;
        }
        printf("Enter the data to be deleted : ");
        scanf("%d",&data);
        t1 = root;
        t2 = root;
        search1(root , data);
}

void preorder(struct  btnode *t)
{
        if(root == NULL)
        {
                printf("No elements in a tree to display");
                return;
        }
        printf("%d-> ",t->value);
        if(t->l != NULL)
        {
                preorder(t->l);
        }
        if(t->r != NULL)
        {
                preorder(t->r);
        }
```

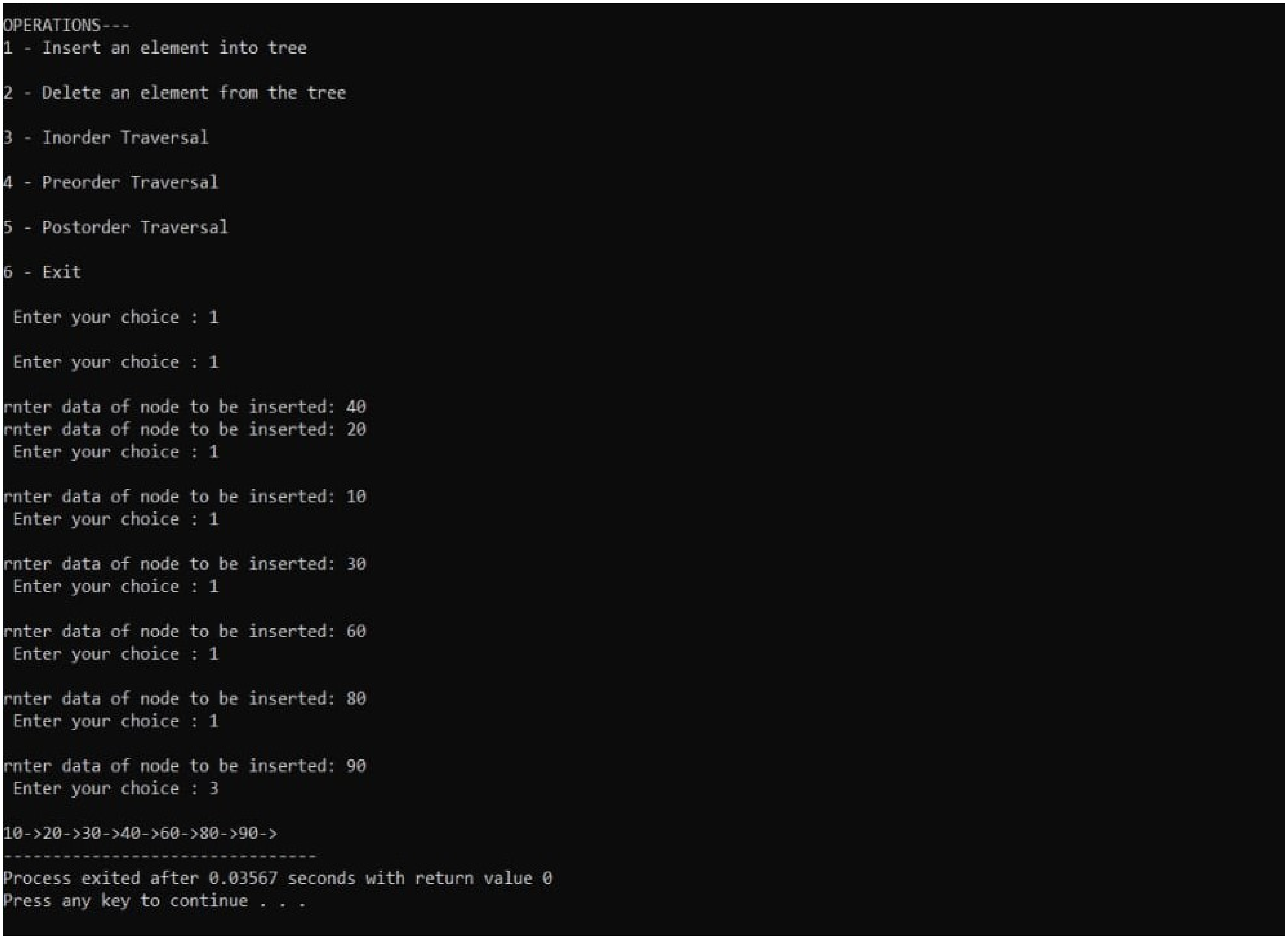
```
}


void postorder(struct btnode *t)
{
        if(root == NULL)
        {
        printf("No elements in a tree to display");
        return;
        }
        if(t->l != NULL)
        {
                postorder(t->l);
        }
        if(t->r != NULL)
        {
                postorder(t->r);
        }
    printf("%d-> ",t->value);
}


void search1(struct btnode *t,int data)
{
        if((data>t->value))
        {
                t1 = t;
                search1(t->r,data);
        }
        else if((data < t->value))
        {
                t1 = t;
                search1(t->l,data);
        }
        else if((data == t->value))
```

```
{
                delete(t);
        }
}


void delete1(struct btnode *t)
{
        int k;
        if((t->l == NULL)&&(t->r == NULL))
        {
                if(t1->l == t)
                {
                        t1->l = NULL;
                }
                else
                {
                        t1->r = NULL;
                }
                t=NULL;
                free(t);
                return;
        }
        else if(t->r == NULL)
        {
                if(t1 == t)
                {
                        root = t->l;
                        t1 = root;
                }
                else if(t1->l == t)
                {
                        t1->l = t->l;
                }
```

```
else
        {
            t1->r = t->l;
        }
        t = NULL;
        free(t);
        return;
    }
    else if(t->l == NULL)
    {
        if(t1 == t)
        {
            root == t->r;
            t1 = root;
        }
        else if(t1->r == t)
        {
            t1->l = t->r;
        }
        else
        {
            t1->l = t->r;
        }
        t == NULL;
        free(t);
        return;
    }

    else if((t->l != NULL)&&(t->r != NULL))
    {
        t2 = root;
        if(t->r != NULL)
        {
```

```
k = smallest(t->r);
            flag =1;
        }
        else
        {
            k = largest(t->l);
            flag = 2;
        }
        search1(root , k);
        t->value = k;
    }
}



int smallest(struct btnode *t)
{
    t2 = t;
    if(t->l != NULL)
    {
        t2 = t;
        return(smallest(t->l));
    }
    else
    {
        return(t->value);
    }
}


int largest(struct btnode *t)
{
    if(t->r != NULL)
    {
        t2 = t;
        return(largest(t->r));
    }
    else
    {
        return(t->value);
    }
}
```

OUTPUT:-



```
OPERATIONS---
1 - Insert an element into tree

2 - Delete an element from the tree

3 - Inorder Traversal

4 - Preorder Traversal

5 - Postorder Traversal

6 - Exit

 Enter your choice : 1

 Enter your choice : 1

rnter data of node to be inserted: 40
rnter data of node to be inserted: 20
 Enter your choice : 1

rnter data of node to be inserted: 10
 Enter your choice : 1

rnter data of node to be inserted: 30
 Enter your choice : 1

rnter data of node to be inserted: 60
 Enter your choice : 1

rnter data of node to be inserted: 80
 Enter your choice : 1

rnter data of node to be inserted: 90
 Enter your choice : 3

10->20->30->40->60->80->90->
--------------------------------
Process exited after 0.03567 seconds with return value 0
Press any key to continue . . .
```