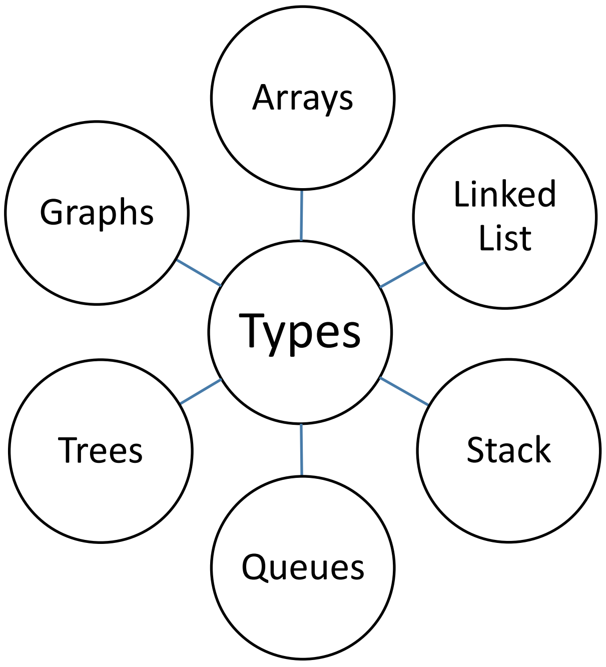
Introduction to Data Structure ?

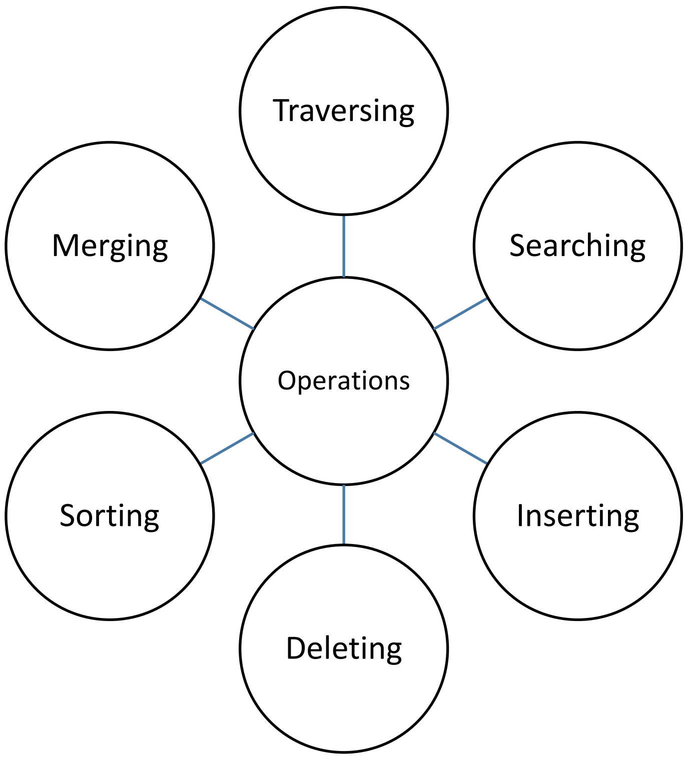
1.1. Definition :

The logical or mathematical model of a particular organization of data is called **DATA STRUCTURE**.

1.2. Types of Data Structure:



1.3. Operation on Data Structure:



ARRAYS

A **linear array** is a list of a finite number n of homogeneous data elements (i.e. of same **data type**) such that:

i. The elements are stored respectively **successive memory locations.**

ii. The elements of array are reference by an **index.**

Importance of Array:

- Code Optimization (less code)
- Random Access
- Easy to traverse data
- Easy to manipulate data
- Easy to sort data etc.

LINKED LISTS

A **linked lists** is a linear collection of data elements, called nodes, where the linear order is given by the means of pointer. And **node** are divided into two parts: the **first part** contain **information of element** and **other part** contains **address of next node** called **link field** or **nextpointer field**.

Importance of Linked Lists:

- Linked List *allows* storage of *duplicate elements* in it.
- Linked List *maintains an order* in which elements are *inserted* in it.
- Linked List are **dynamic data structures.**
- Linked List have **effective memory utilization.**

STACK

A **stack** is a list of elements in which an element may be inserted or deleted only at one end, called the **top** of the stack.

Two basic operations are associated with the **stack**:

- ➔ **PUSH** means inserting element in stack
- ➔ **POP** means deleting element from stack

Importance of Stack:

- ➔ A stack is a container of objects that are inserted and removed according to the **Last-IN First-Out** (**LIFO**).
- ➔ A stack is a limited access data structure.
- ➔ Elements can be added and removed from the Stack only at the TOP.

QUEUES

A queues is a linear list of elements in which deletion can take place only at one end called the front and the insertion can take place only at the other end called the rear. It is also called **First-In First-Out** (**FIFO**).

Importance of Queues:

- ➔ It is basically have infinite length as compared to array because array have finite length.
- ➔ It is fast and Flexible.

TREES

Unlike Arrays, Linked Lists, Stack and queues, which are linear data structures, **TREES** are hierarchical data structures.

Importance of Tress:

- ➔ One reason to use trees might be because you want to store information that naturally forms a hierarchy. For example, the file system on a computer
- ➔ Trees (with some ordering e.g., BST) provide moderate access/search (quicker than Linked List and slower than arrays).
- ➔ Trees provide moderate insertion/deletion (quicker than Arrays and slower than Unordered Linked Lists).
- ➔ Like Linked Lists and unlike Arrays, Trees don't have an upper limit on number of nodes as nodes are linked using pointers.
- ➔ Manipulate hierarchical data.
- ➔ Make information easy to search (see tree traversal).
- ➔ Manipulate sorted lists of data.
- ➔ As a workflow for compositing digital images for visual effects.
- ➔ Router algorithms
- ➔ Form of a multi-stage decision-making (see business chess).

GRAPHS

A Graph is a non-linear data structure consisting of nodes and edges. The nodes are sometimes also referred to as vertices and the edges are lines or arcs that connect any two nodes in the graph. More formally a Graph can be defined as,

A Graph consists of a finite set of vertices(or nodes) and set of Edges which connect a pair of nodes.

Importance of Graphs:

- ➔Graphs are used to solve many real-life problems.
- ➔Graphs are used to represent networks. The networks may include paths in a city or telephone network or circuit network.
- ➔Graphs are also used in social networks like linkedIn, Facebook. For example, in Facebook, each person is represented with a vertex(or node).
- ➔Each node is a structure and contains information like person id, name, gender, locale etc.

Experiment 1: Program to insert a new element

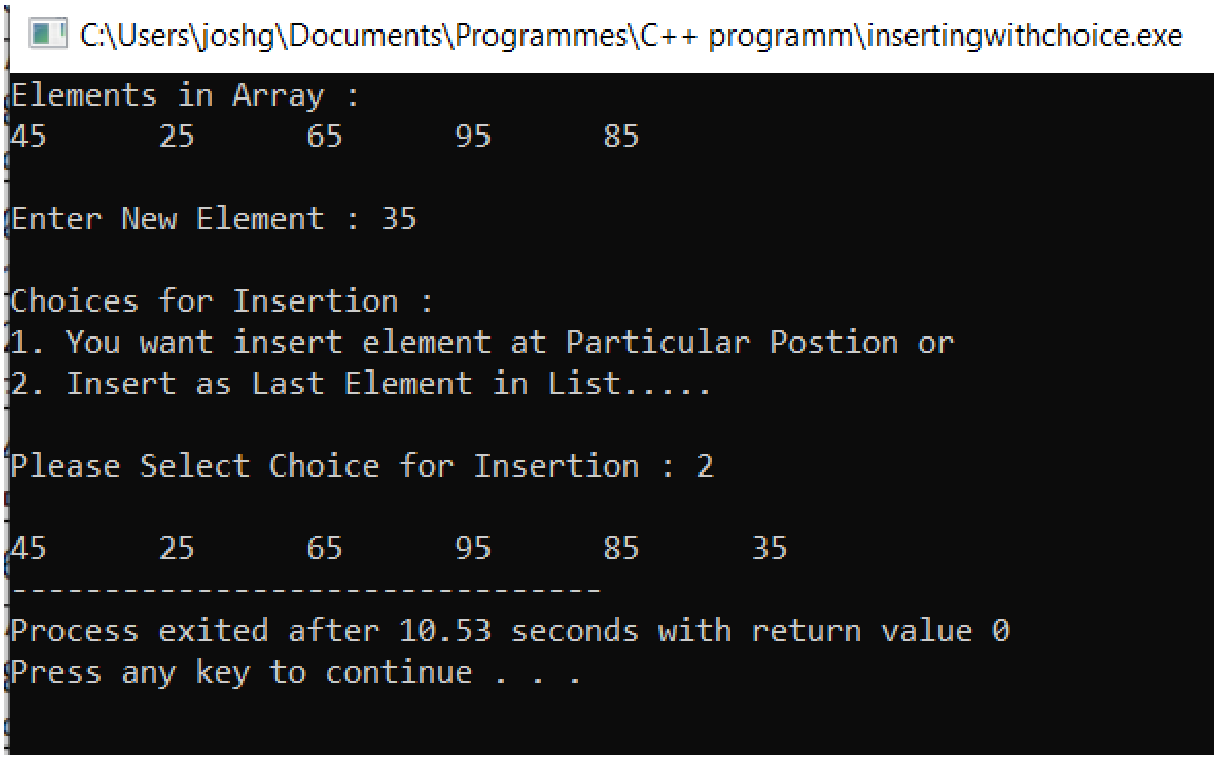
(i) **at end as well as**
(ii) **at a given position in an array**

Code :

```
#include<iostream>

using namespace std;

int main(){

	int size=5;

	int arr[size] = {45,25,65,95,85};

	cout << "Elements in Array : \n";

	for(int i=0;i<size;i++){

		cout << arr[i] << "\t";

	}

	cout << endl << endl;

	int newElement;

	cout << "Enter New Element : ";

	cin >> newElement;

	cout << endl;

	cout << "Choices for Insertion : \n";
```

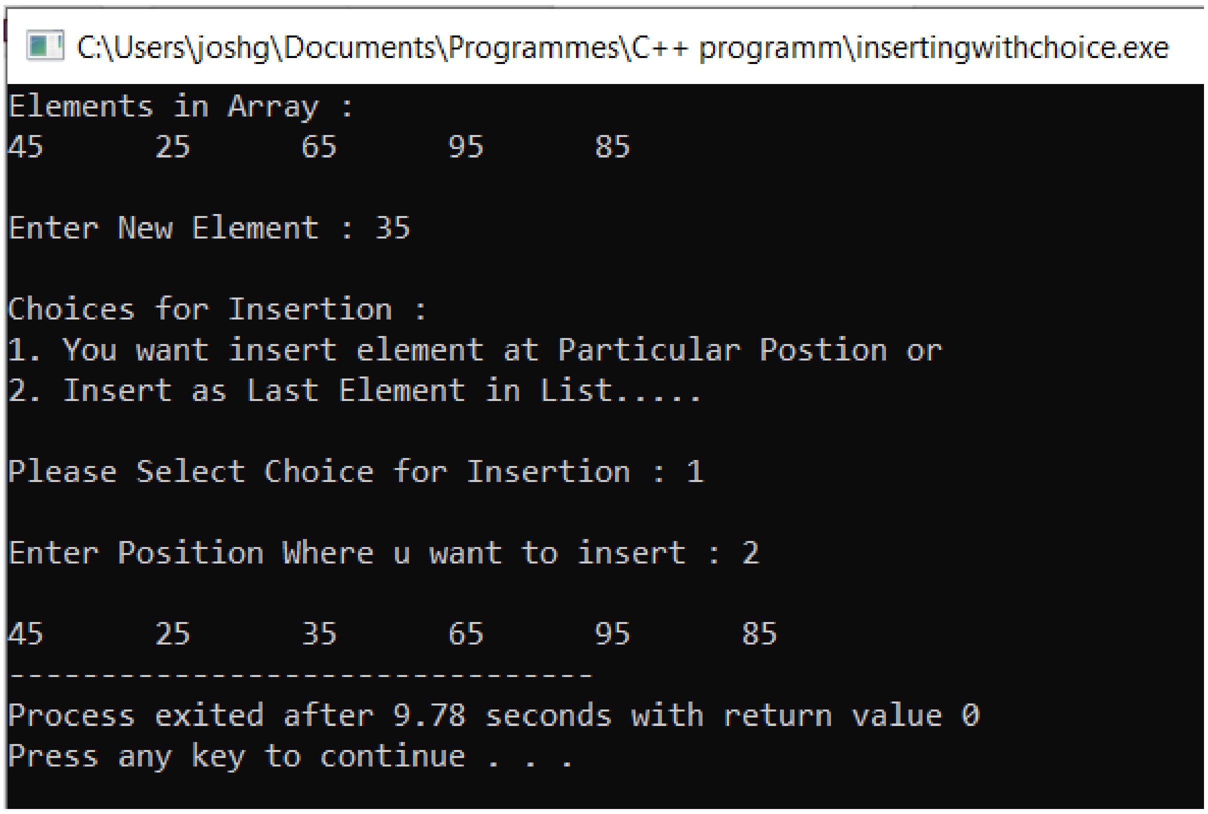
```
    cout << "1. You want insert element at Particular
Postion or \n";
    cout << "2. Insert as Last Element in List.....
\n"<<endl;
    int choice;
    cout << "Please Select Choice for Insertion : ";
    cin >> choice;
    cout << endl;
    if(choice==1){
        int position;
        cout << "Enter Position Where u want to insert :
";
        cin >> position;
        cout << endl;
        int  j =size;
        for(j=size;j>=position;j--){
            arr[j+1] = arr[j];
        }
        arr[position] = newElement;
        size = size+1;
        for(int i=0;i<size;i++){
            cout << arr[i] << "\t";
        }
    }
    else if(choice==2){
        arr[size] = newElement;
        size = size+1;
        for(int i=0;i<size;i++){
            cout << arr[i] << "\t";
        }
    }
    else
        cout << "You Have not entered any Choice";
}
```

Output : (I) Element inserted at last


C:\Users\joshg\Documents\Programmes\C++ programm\insertingwithchoice.exe

```
Elements in Array :
45      25      65      95      85

Enter New Element : 35

Choices for Insertion :
1. You want insert element at Particular Postion or
2. Insert as Last Element in List.....

Please Select Choice for Insertion : 2

45      25      65      95      85      35
--------------------------------
Process exited after 10.53 seconds with return value 0
Press any key to continue . . .
```

(II) Element inserted at particular position


C:\Users\joshg\Documents\Programmes\C++ programm\insertingwithchoice.exe

```
Elements in Array :
45      25      65      95      85

Enter New Element : 35

Choices for Insertion :
1. You want insert element at Particular Postion or
2. Insert as Last Element in List.....

Please Select Choice for Insertion : 1

Enter Position Where u want to insert : 2

45      25      35      65      95      85
--------------------------------
Process exited after 9.78 seconds with return value 0
Press any key to continue . . .
```

Experiment 2: Program to delete an element

(I) from a given whose value is given
(II) or whose position is given.

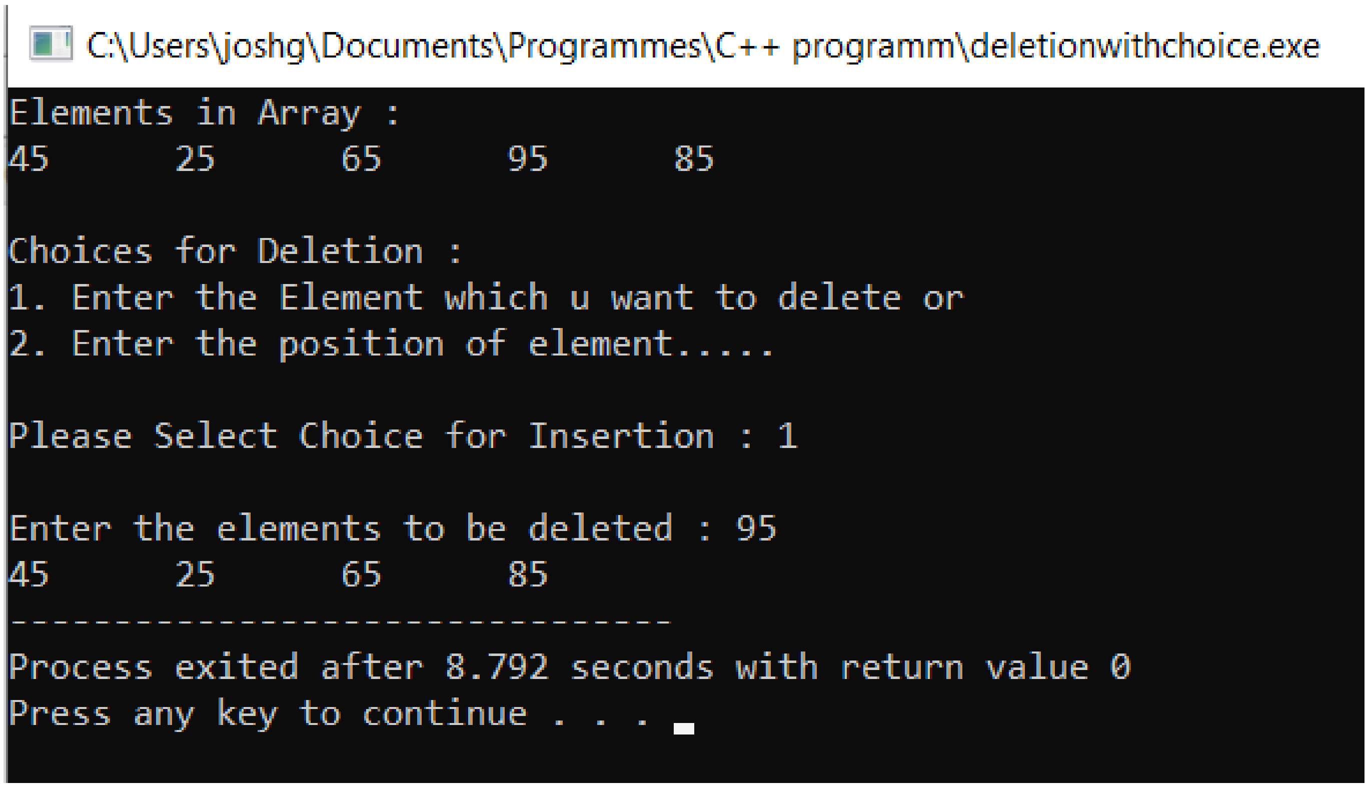
Code :

```
#include<iostream>

using namespace std;

int main(){

	int size=5;

	int arr[size] = {45,25,65,95,85};

	cout << "Elements in Array : \n";

	for(int i=0;i<size;i++){

		cout << arr[i] << "\t";

	}

	cout << endl << endl;

	cout << "Choices for Deletion : \n";

	cout << "1. Enter the Element which u want to delete 
or \n";

	cout << "2. Enter the position of element.....
\n"<<endl;

	int choice;
```

```
cout << "Please Select Choice for Insertion : ";
cin >> choice;
cout << endl;
if(choice==1){
    int delElement;
    cout << "Enter the elements to be deleted : ";
    cin >> delElement;
    for(int i=0;i<size;i++){
        if(arr[i]==delElement){
          for(int j=i;j<size;j++){
            arr[j]=arr[j+1];
            }
            break;
    }
}
for(int i=0;i<(size-1);i++)
  {
    cout<<arr[i]<<"\t";
  }
```

```
	}
	else if(choice==2){
		int position;
		cout << "Enter position of element u want to
delete : ";
		cin >> position;
		cout <<endl;
		int item = arr[position];
		int j = size;
		for(j=position;j<size;j++){
			arr[j] = arr[j+1];
		}
		size = size-1;
		for(int i=0;i<size;i++){
			cout << arr[i] << "\t";
		}
	}
	else
		cout << "You Have not entered any Choice";
}
```

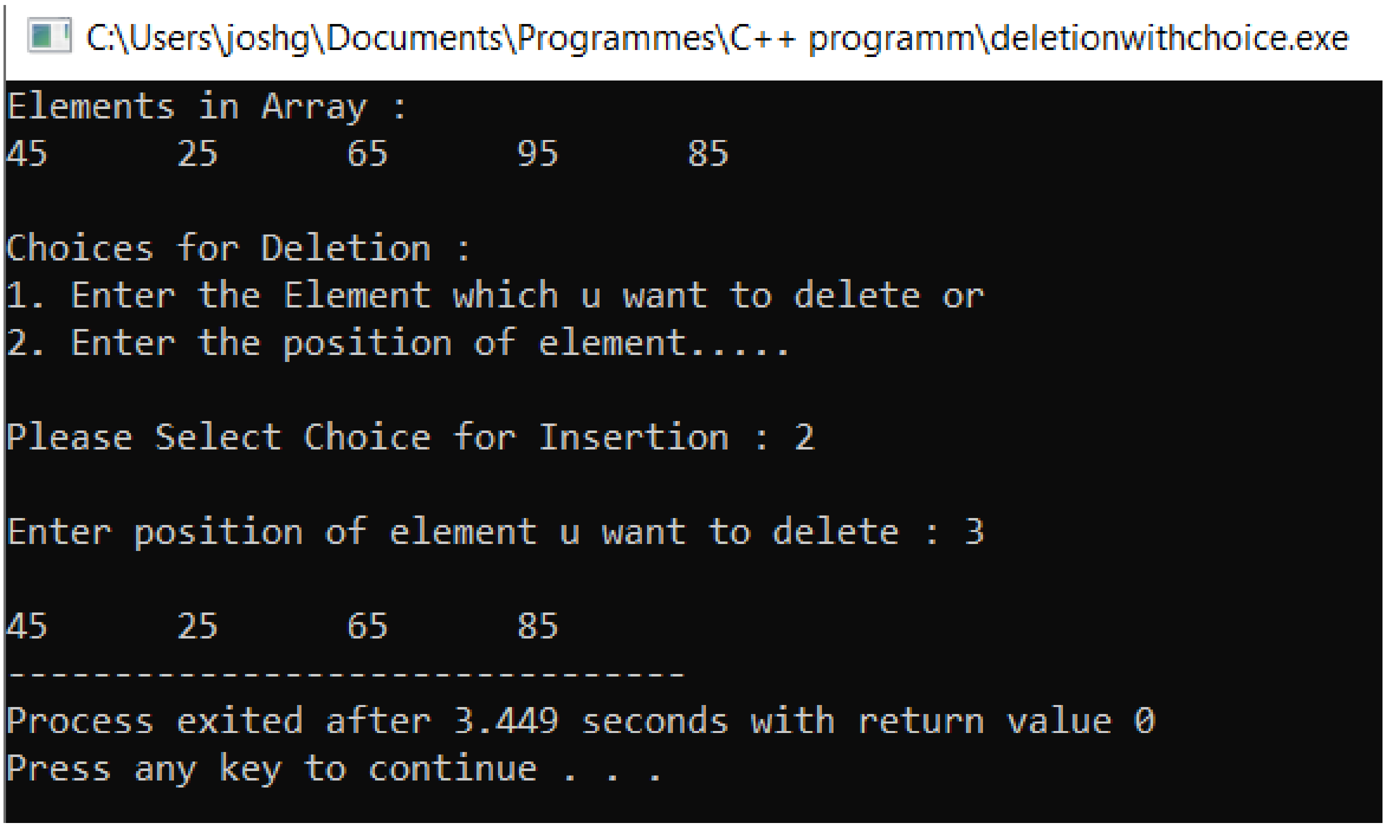
Output: (I) Element whose value is given



C:\Users\joshg\Documents\Programmes\C++ programm\deletionwithchoice.exe

```
Elements in Array :
45      25      65      95      85

Choices for Deletion :
1. Enter the Element which u want to delete or
2. Enter the position of element.....

Please Select Choice for Insertion : 1

Enter the elements to be deleted : 95
45      25      65      85
--------------------------------
Process exited after 8.792 seconds with return value 0
Press any key to continue . . .
```

(II) Element whose position is given



C:\Users\joshg\Documents\Programmes\C++ programm\deletionwithchoice.exe

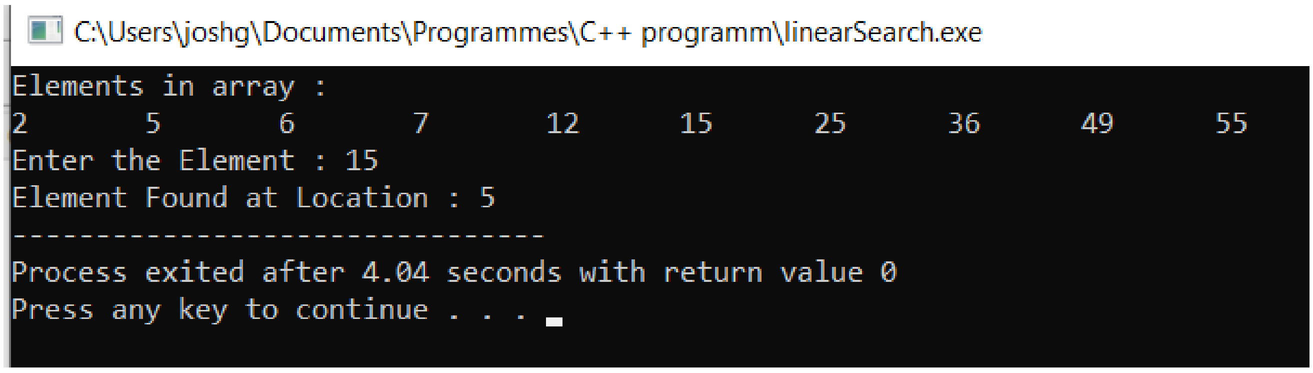
```
Elements in Array :
45      25      65      95      85

Choices for Deletion :
1. Enter the Element which u want to delete or
2. Enter the position of element.....

Please Select Choice for Insertion : 2

Enter position of element u want to delete : 3

45      25      65      85
--------------------------------
Process exited after 3.449 seconds with return value 0
Press any key to continue . . .
```

Experiment 3: Program to find the location of a given element using Linear Search

Code :

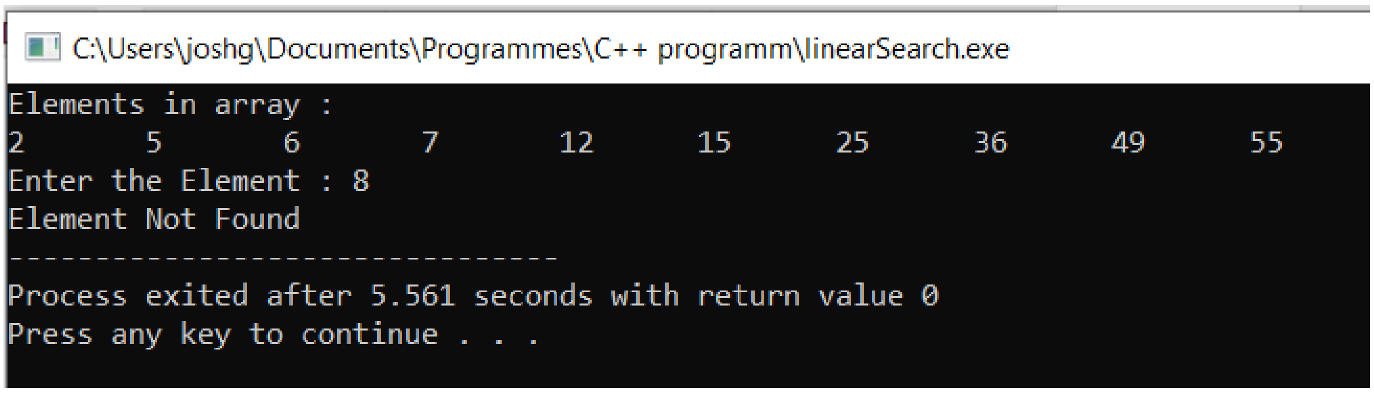
```
#include<iostream>

using namespace std;

int linearsearch(int arr[],int size, int searchElement){

    for(int i=0;i<=size-1;i++)

        if(arr[i]==searchElement)

            return i;


    return -1;

}

int main(){

    int size = 10;

    int arr[size] = {2,5,6,7,12,15,25,36,49,55};

    cout << "Elements in array : \n";

    for(int i=0;i<10;i++){

        cout << arr[i] << "\t";

    }
    int searchElement;
    cout << "\nEnter the Element : ";
    cin >> searchElement;
    int mid = linearsearch(arr,size,searchElement);
    if(mid== -1)
        cout << "Element Not Found";
    else
        cout << "Element Found at Location : " << mid;
}
```

Output: (I.) Element Found At Particular Position


C:\Users\joshg\Documents\Programmes\C++ programm\linearSearch.exe

```
Elements in array :
2       5       6       7       12      15      25      36      49      55
Enter the Element : 15
Element Found at Location : 5
--------------------------------
Process exited after 4.04 seconds with return value 0
Press any key to continue . . .
```

(II.) Element Not Found In Array


C:\Users\joshg\Documents\Programmes\C++ programm\linearSearch.exe

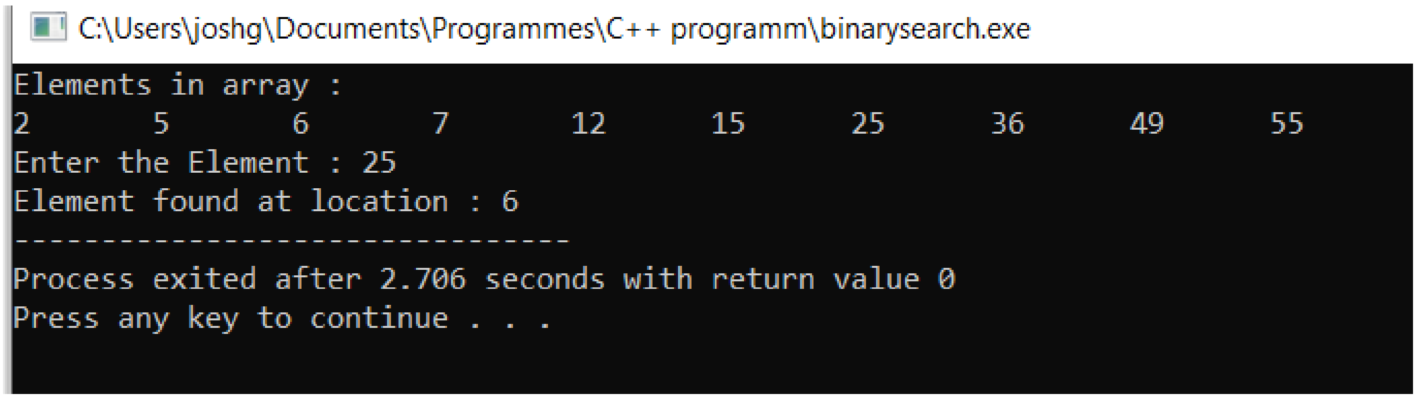
```
Elements in array :
2       5       6       7       12      15      25      36      49      55
Enter the Element : 8
Element Not Found
--------------------------------
Process exited after 5.561 seconds with return value 0
Press any key to continue . . .
```

Experiment 4: Program to find the location of a given element using Binary Search

Code :

```
#include<iostream>
using namespace std;
int binarysearch(int arr[],int beg, int end, int
searchElement){
	int mid = (beg+end)/2;
	if(arr[mid] == searchElement){
		return mid;
	}
	if(arr[mid] > searchElement){
		return binarysearch(arr, beg, mid-1,
searchElement);
	}
	else{
		return binarysearch(arr, mid+1, end,
searchElement);
	}

	return -1;
}
int main(){
    int size = 10;
    int arr[size] = {2,5,6,7,12,15,25,36,49,55};
    //int n = sizeof(arr)/sizeof(arr[0]);
    cout << "Elements in array : \n";
    for(int i=0;i<10;i++){
        cout << arr[i] << "\t";
    }
    int searchElement;
    cout << "\nEnter the Element : ";
    cin >> searchElement;
    int mid = binarysearch(arr,0,size,searchElement);
    cout << "Element found at location : " << mid;
}
```

Output:



C:\Users\joshg\Documents\Programmes\C++ programm\binarysearch.exe

```
Elements in array :
2       5       6       7       12      15      25      36      49      55
Enter the Element : 25
Element found at location : 6
--------------------------------
Process exited after 2.706 seconds with return value 0
Press any key to continue . . .
```

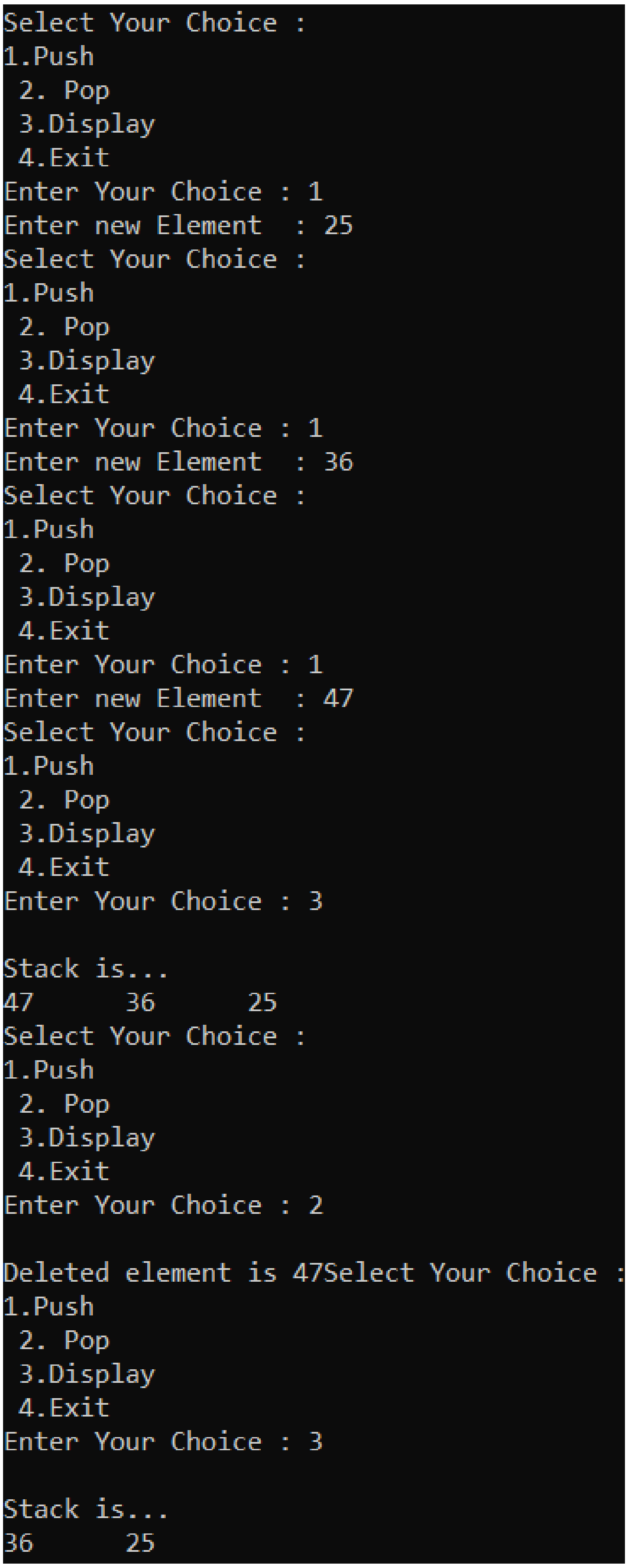
Experiment 5: Program to implement push and pop operations on a stack using linear array

Code:

```
#include<iostream>
#define size 5
using namespace std;
int top=-1, stack[size];
void push(int newElement){
        if(top==size-1){
                cout << "Stack is Full";
        }
        else{
                top=top+1;
                stack[top]= newElement;
        }
}

void pop()
{
        if(top==-1){
                cout << "\nStack is empty!!";
        }
        else{
                cout <<"\nDeleted element is " << stack[top];
                top=top-1;
        }
}

void display(){
        int i;
        if(top==-1){
                cout <<"\nStack is empty!!";
        }
        else{
                cout << "\nStack is...\n";
                for(i=top;i>=0;--i)
                        cout << stack[i] <<"\t";
        }
        cout << endl;
}
int main(){
        int choice;
        int newElement;
        while(1){
                cout << "Select Your Choice : "<<endl;
                cout << "1.Push \n 2. Pop\n 3.Display\n 4.Exit" <<
endl;
```

```
            cout << "Enter Your Choice : ";
            cin >> choice;
            cout << endl;
            switch(choice){
                case 1: {
                    cout << "Enter new Element  : ";
                    cin >> newElement;
                    push(newElement);
                    break;
                }
                case 2: pop();
                    break;
                case 3: display();
                    break;
                case 4: exit(0);
                    break;
                default : cout << "Invalid Choice";
                    break;
            }
        }
    }
```

Output:



```
Select Your Choice :
1.Push
 2. Pop
 3.Display
 4.Exit
Enter Your Choice : 1
Enter new Element  : 25
Select Your Choice :
1.Push
 2. Pop
 3.Display
 4.Exit
Enter Your Choice : 1
Enter new Element  : 36
Select Your Choice :
1.Push
 2. Pop
 3.Display
 4.Exit
Enter Your Choice : 1
Enter new Element  : 47
Select Your Choice :
1.Push
 2. Pop
 3.Display
 4.Exit
Enter Your Choice : 3

Stack is...
47      36      25
Select Your Choice :
1.Push
 2. Pop
 3.Display
 4.Exit
Enter Your Choice : 2

Deleted element is 47Select Your Choice :
1.Push
 2. Pop
 3.Display
 4.Exit
Enter Your Choice : 3

Stack is...
36      25
```

Experiment 6: Program to convert an infix expression to a postfix expression using stacks.

Code:

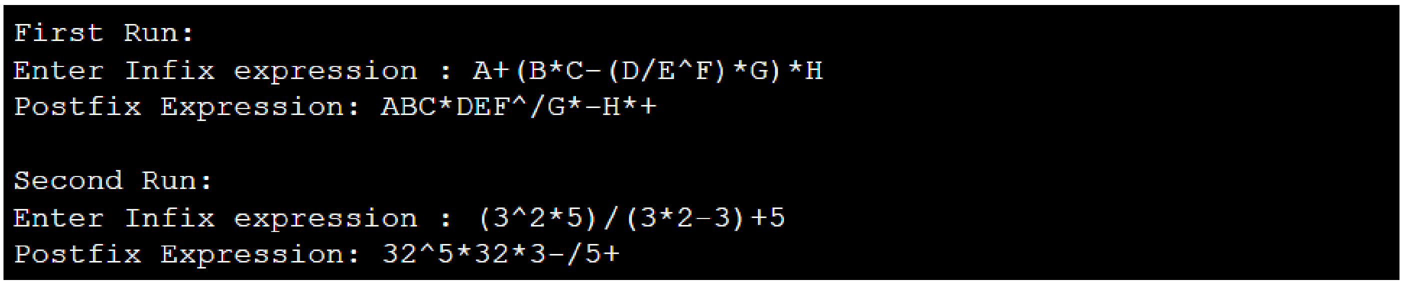
```
#include<stdio.h>
#include<stdlib.h>      /* for exit() */
#include<ctype.h>     /* for isdigit(char ) */
#include<string.h>
#define SIZE 100
char stack[SIZE];
int top = -1;
void push(char item){
        if(top >= SIZE-1){
                printf("\nStack Overflow.");
        }
        else{
                top = top+1;
                stack[top] = item;
        }
}
char pop(){
        char item ;
        if(top <0)
        {
                printf("stack under flow: invalid infix expression");
                getchar();
                exit(1);
        }
        else{
                item = stack[top];
                top = top-1;
                return(item);
        }
}
int is_operator(char symbol){
        if(symbol == '^' || symbol == '*' || symbol == '/' || symbol == '+'
|| symbol =='-'){
                return 1;
        }
        else{
        return 0;
        }
}

int precedence(char symbol){
        if(symbol == '^'){
                return(3);
        }
        else if(symbol == '*' || symbol == '/'){
                return(2);
        }
        else if(symbol == '+' || symbol == '-'){
```

```
		return(1);
	}
	else{
		return(0);
	}
}
void InfixToPostfix(char infix_exp[], char postfix_exp[]){
	int i, j;
	char item;
	char x;
	push('(');
	strcat(infix_exp,")");
	i=0;
	j=0;
	item=infix_exp[i];
	while(item != '\0'){
		if(item == '('){
			push(item);
		}
		else if( isdigit(item) || isalpha(item)){
			postfix_exp[j] = item;
			j++;
		}
		else if(is_operator(item) == 1){
			x=pop();
```

```
				while(is_operator(x) == 1 && precedence(x)>=
precedence(item)){
					postfix_exp[j] = x;
					j++;
					x = pop();
				}
				push(x);
				push(item);
			}
			else if(item == ')'){
				x = pop();
				while(x != '('){
					postfix_exp[j] = x;
					j++;
					x = pop();
				}
			}
			else{
				printf("\nInvalid infix Expression.\n");
				getchar();
				exit(1);
			}
			i++;
			item = infix_exp[i];
		}
		if(top>0){
```

```
                printf("\nInvalid infix Expression.\n");
                getchar();
                exit(1);
        }
        postfix_exp[j] = '\0';
}
int main(){
        char infix[SIZE], postfix[SIZE];
        printf("\nEnter Infix expression : ");
        gets(infix);

        InfixToPostfix(infix,postfix);
        printf("Postfix Expression: ");
        puts(postfix);

        return 0;
}
```

Output:



```
First Run:
Enter Infix expression : A+(B*C-(D/E^F)*G)*H
Postfix Expression: ABC*DEF^/G*-H*+

Second Run:
Enter Infix expression : (3^2*5)/(3*2-3)+5
Postfix Expression: 32^5*32*3-/5+
```

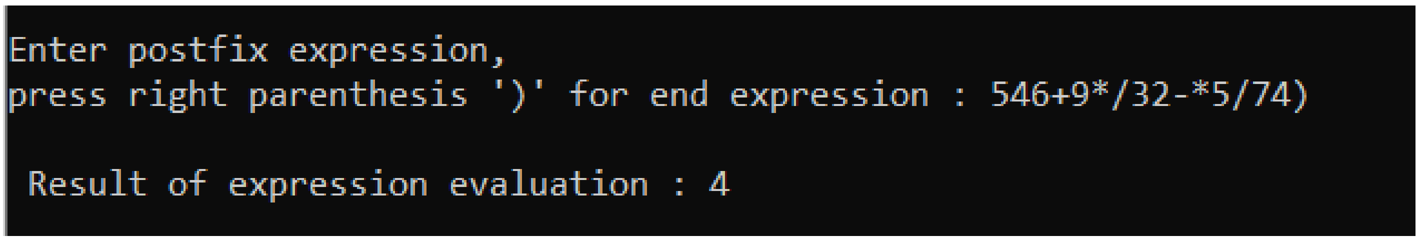
Experiment 7: Program to evaluate a postfix expression using stacks.

Code:

```
#include<stdio.h>
 #include<ctype.h>
 # define MAXSTACK 100
 # define POSTFIXSIZE 100
 int stack[MAXSTACK];
 int top = -1 ;
 void push(int item){
        if(top >= MAXSTACK -1){
               printf("stack over flow");
               return;
        }
        else{
               top = top + 1 ;
               stack[top]= item;
        }
 }
 int pop(){
        int item;
        if(top <0){
              printf("stack under flow");
        }
        else{
               item = stack[top];
               top = top - 1;
               return item;
        }
}
void EvalPostfix(char postfix[]){
      int i ;
      char ch;
      int val;
      int A, B ;

      for (i = 0 ; postfix[i] != ')'; i++){
             ch = postfix[i];
             if (isdigit(ch)){
                    push(ch - '0');
             }
             else if (ch == '+' || ch == '-' || ch == '*' || ch == '/'){
                    A = pop();
                    B = pop();

                    switch (ch){
```

```
                    case '*':
                    val = B * A;
                    break;

                    case '/':
                    val = B / A;
                    break;

                    case '+':
                    val = B + A;
                    break;

                    case '-':
                    val = B - A;
                    break;
                }
                push(val);
            }
        }
        printf( " \n Result of expression evaluation : %d \n", pop()) ;
}

int main(){
        int i ;
        char postfix[POSTFIXSIZE];
        printf("ASSUMPTION: There are only four operators(*, /, +,
-) in an expression and operand is single digit only.\n");
        printf( " \nEnter postfix expression,\npress right parenthesis
')' for end expression : ");

        for (i = 0 ; i <= POSTFIXSIZE - 1 ; i++)
        {
                scanf("%c", &postfix[i]);

                if ( postfix[i] == ')' )
                { break; }
        }
        EvalPostfix(postfix);

        return 0;
 }
```

Output:



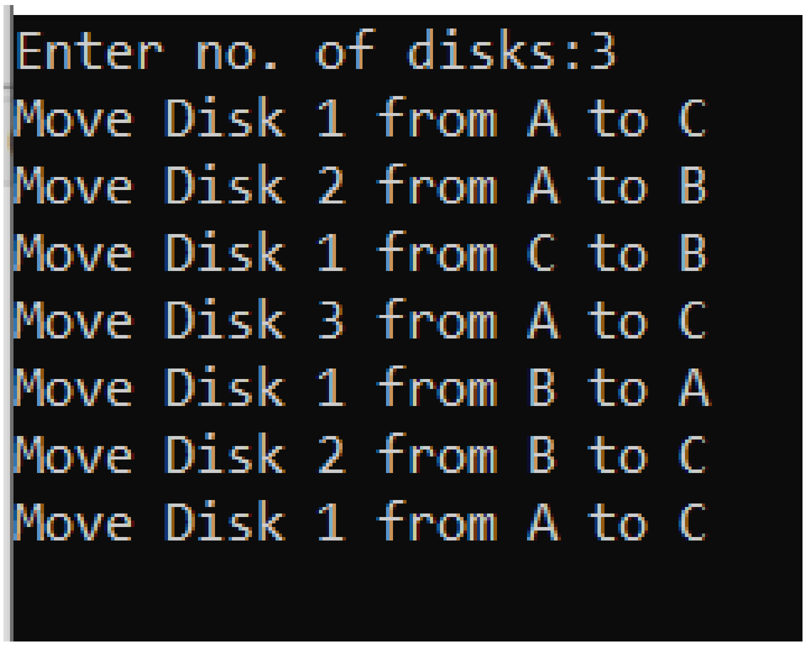
```
Enter postfix expression,
press right parenthesis ')' for end expression : 546+9*/32-*5/74)

 Result of expression evaluation : 4
```

Experiment 8: Implement recursive function for Tower of Hanoi problem.

Code:

```
#include<iostream>
using namespace std;

void TOH(int n,char Beg, char Aux,char End){
	if(n==1){
		cout<<"Move Disk "<<n<<" from 
"<<Beg<<" to "<<End<<endl;
		return;
	}
	else{
		TOH(n-1,Beg,End,Aux);
		cout<<"Move Disk "<<n<<" from 
"<<Beg<<" to "<<End<<endl;
		TOH(n-1,Aux,Beg,End);
	}
}

int main(){
    int n;
    cout<<"Enter no. of disks:";
    cin>>n;
    TOH(n,'A','B','C');
    return 0;
}
```

Output:



```
Enter no. of disks:3
Move Disk 1 from A to C
Move Disk 2 from A to B
Move Disk 1 from C to B
Move Disk 3 from A to C
Move Disk 1 from B to A
Move Disk 2 from B to C
Move Disk 1 from A to C
```

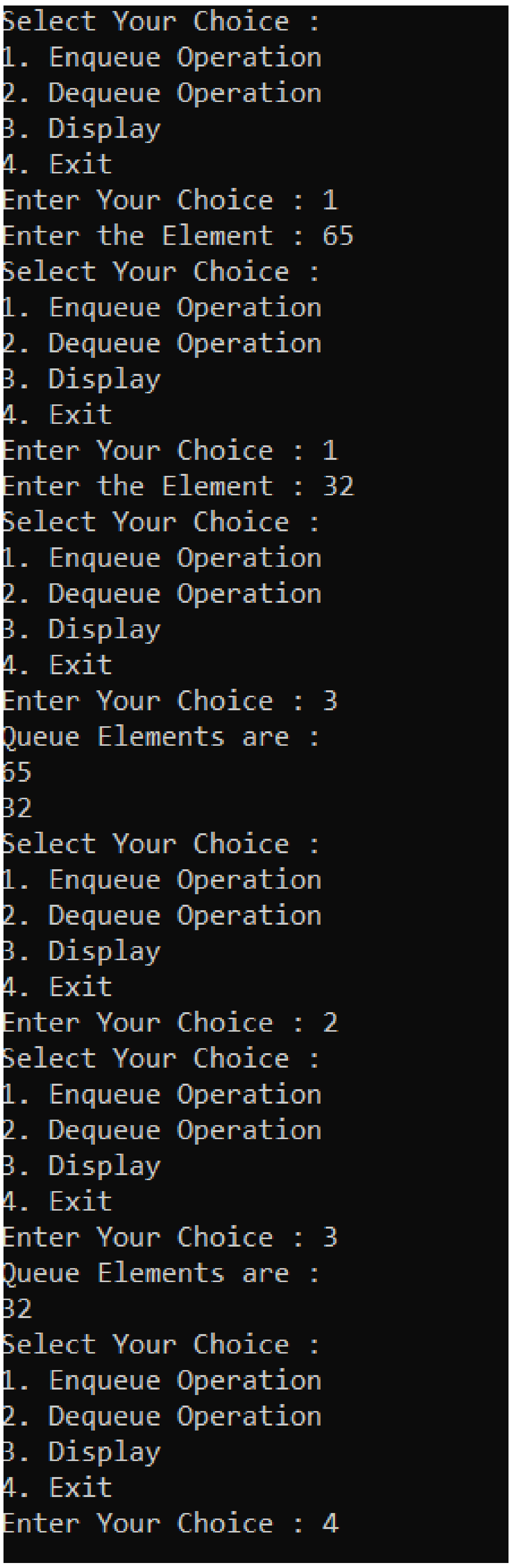
Experiment 9: Program to implement insertion and delete on operations in a queue using linear array.

Code:

```
 #include<iostream>
using namespace std;
int queue[100],n=100,front=-1,rear=-1;
void enqueue(){
    int new_element;
    cout << "Enter the Element : ";
    cin >> new_element;
    if(front==rear+1)
        cout << "Queue Is Full";
    else if(front == -1){
        front = 0;
        rear = 0;
    }
    else
        rear = rear + 1;

    queue[rear] = new_element;
}

void dequeue(){
    int new_element;
    if(front==-1)
        cout << "Queue Is UnderFlow";

    new_element = queue[front];
    if(front==rear){
        front = 0;
        rear = 0;
    }
    else
        front++;
}
void dispaly(){
    if(front == -1)
        cout << "Queue is Empty";
    else {
        cout << "Queue Elements are : " << endl;
        for(int i=front;i<=rear;i++){
            cout << queue[i] << " ";
            cout << endl;
        }
    }
}
```

```
int main(){
    int operation_choice;
while(1){
        cout << "Select Your Choice : " << endl;
        cout << "1. Enqueue Operation" << endl;
        cout << "2. Dequeue Operation" << endl;
        cout << "3. Display" << endl;
        cout << "4. Exit" << endl;
        cout << "Enter Your Choice : ";
        cin >> operation_choice;
        switch (operation_choice){
        case 1: enqueue();
            break;
        case 2: dequeue();
            break;
        case 3: dispaly();
            break;
        case 4: exit(0);
            break;
        default: printf("Invalid Choice!!\n");
            break;
        }
    }
    return 0;
}
```

Output:



```
Select Your Choice :
1. Enqueue Operation
2. Dequeue Operation
3. Display
4. Exit
Enter Your Choice : 1
Enter the Element : 65
Select Your Choice :
1. Enqueue Operation
2. Dequeue Operation
3. Display
4. Exit
Enter Your Choice : 1
Enter the Element : 32
Select Your Choice :
1. Enqueue Operation
2. Dequeue Operation
3. Display
4. Exit
Enter Your Choice : 3
Queue Elements are :
65
32
Select Your Choice :
1. Enqueue Operation
2. Dequeue Operation
3. Display
4. Exit
Enter Your Choice : 2
Select Your Choice :
1. Enqueue Operation
2. Dequeue Operation
3. Display
4. Exit
Enter Your Choice : 3
Queue Elements are :
32
Select Your Choice :
1. Enqueue Operation
2. Dequeue Operation
3. Display
4. Exit
Enter Your Choice : 4
```

Experiment 10: Program to implement linked list

Code:

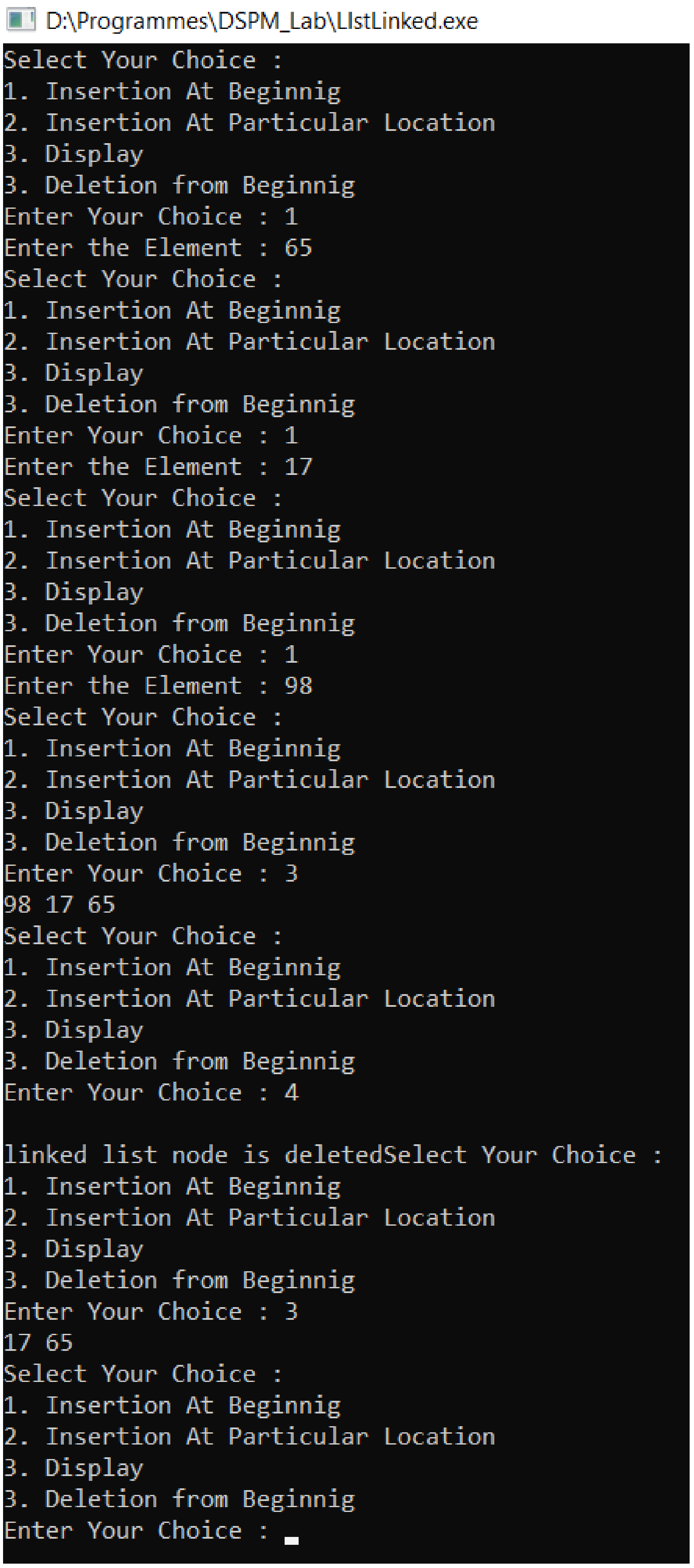
```
#include<stdio.h>
#include<stdlib.h>
struct node{
    int data;
    struct node* next;
};
struct node* start = NULL;
void InsertionAtBeginnig(int data){
    struct node* temp;
    temp = (struct node*)malloc(sizeof(struct node));

    if(start==NULL){
        start = temp;
        start->data = data;
        start->next = NULL;
        return;
    }
    else{
        temp->data = data;
        temp->next = start;
        start = temp;
    }
}
void InsertionAtParticularLoc(int data,int loc){
    struct node *temp1;
    //struct node* temp2;
    temp1 = (struct node*)malloc(sizeof(struct node));
    temp1->data = data;
    struct node* temp2 = start;
    if(loc == 1){
        //temp2->data = data;
        temp2->next = temp1;
    }
    else{
        for(int i=0;i<(loc-1);i++){
            temp2 = temp2->next;
        }
        temp1->next = temp2->next;
        temp2->next = temp1;
    }
}
void delete_beg(){
      int n;
      struct node *temp;
```

```
        temp=(struct node*)malloc(sizeof(struct node));
        if(start==NULL){
        printf("\nlinked list is empty");
        }
        else{
                n=start->data;
                temp=start->next;
                free(start);
                start=temp;
                printf("\nlinked list node is deleted");
        }
}
void display()
{
     struct node* temp = start;
     while(temp != NULL){
        printf("%d ", temp->data);
        temp = temp->next;
     }
     printf("\n");
}
int main(){
     int operation_choice;
```

```
while(1){
    printf("Select Your Choice : \n");
    printf("1. Insertion At Beginnig \n");
    printf("2. Insertion At Particular Location \n");
    printf("3. Display\n");
    printf("3. Deletion from Beginnig \n");
    printf("Enter Your Choice : ");
    scanf("%d",&operation_choice);
    switch (operation_choice){
    case 1: {
        int element;
        printf("Enter the Element : ");
        scanf("%d",&element);
        InsertionAtBeginnig(element);
        break;
    }
    case 2: {
        int element,postion;
        printf("Enter the Element : ");
        scanf("%d",&element);
        printf("Enter the position : ");
        scanf("%d",&postion);
        InsertionAtParticularLoc(element,postion);
```

```
                break;
            }
            case 3: display();
                break;
            case 4: delete_beg();
                break;
            default: exit(0);
                break;
            }
        }
        return 0;
    }
```

Output:



D:\Programmes\DSPM_Lab\LIstLinked.exe

```
Select Your Choice :
1. Insertion At Beginnig
2. Insertion At Particular Location
3. Display
3. Deletion from Beginnig
Enter Your Choice : 1
Enter the Element : 65
Select Your Choice :
1. Insertion At Beginnig
2. Insertion At Particular Location
3. Display
3. Deletion from Beginnig
Enter Your Choice : 1
Enter the Element : 17
Select Your Choice :
1. Insertion At Beginnig
2. Insertion At Particular Location
3. Display
3. Deletion from Beginnig
Enter Your Choice : 1
Enter the Element : 98
Select Your Choice :
1. Insertion At Beginnig
2. Insertion At Particular Location
3. Display
3. Deletion from Beginnig
Enter Your Choice : 3
98 17 65
Select Your Choice :
1. Insertion At Beginnig
2. Insertion At Particular Location
3. Display
3. Deletion from Beginnig
Enter Your Choice : 4

linked list node is deletedSelect Your Choice :
1. Insertion At Beginnig
2. Insertion At Particular Location
3. Display
3. Deletion from Beginnig
Enter Your Choice : 3
17 65
Select Your Choice :
1. Insertion At Beginnig
2. Insertion At Particular Location
3. Display
3. Deletion from Beginnig
Enter Your Choice : _
```

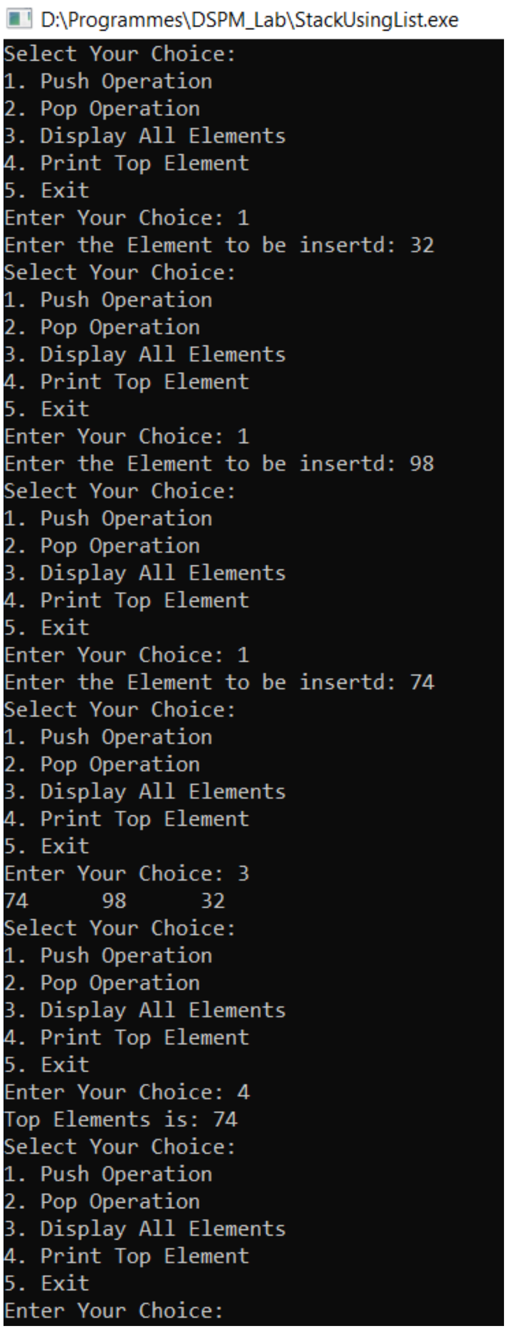
Experiment 11: Program to implement push and pop operations on a stack using linked list

Code:

```
#include<stdio.h>
#include<stdlib.h>
struct stack{
    int data;
    struct stack* next;
};

struct stack* top;
int newNode(int data){
    struct stack* tempNode = (struct stack*)malloc(sizeof(struct
stack));
    tempNode->data = data;
    tempNode->next = top;
    top = tempNode;
    return top;
}
void pushOperation(struct stack* tempNode,int data){

    if(tempNode == NULL){
        return newNode(data);
    }
    else
        return pushOperation(tempNode->next,data);

}

void popOperation(int data){
    struct stack* tempNode;
    if(top == NULL)
        printf("Stack is under flow");
    else{
        tempNode = top;
        top = top->next;
        top->next = NULL;
        free(tempNode);
    }

}
void topElement(){
    printf("Top Elements is: %d\n",top->data);
}

void dispay(){
    struct stack* tempNode = top;
    while(tempNode != NULL){
        printf("%d \t",tempNode->data);
        tempNode = tempNode->next;
    }
    printf("\n");
}

int main(){
    int data,choice;
    while(1){
        printf("Select Your Choice: \n");
        printf("1. Push Operation \n");
        printf("2. Pop Operation \n");
        printf("3. Display All Elements \n");
        printf("4. Print Top Element \n");
        printf("5. Exit \n");
        printf("Enter Your Choice: ");
        scanf("%d",&choice);
        switch(choice){
            case 1: {
                printf("Enter the Element to be insertd: ");
                scanf("%d",&data);
                pushOperation(top,data);
                break;
            }
            case 2: {
                printf("Enter the Element to be deleted: ");
                scanf("%d",&data);
                popOperation(data);
                break;
            }
            case 3: {
                dispay();
                break;
            }
            case 4: {
                topElement();
                break;
            }
            case 5: {
                exit(0);
                break;
            }
            default: {
                printf("Invalid Choice!!");
                break;
```

```
            }
        }
    }

    return 0;
}
```

Output:



```
D:\Programmes\DSPM_Lab\StackUsingList.exe
Select Your Choice:
1. Push Operation
2. Pop Operation
3. Display All Elements
4. Print Top Element
5. Exit
Enter Your Choice: 1
Enter the Element to be insertd: 32
Select Your Choice:
1. Push Operation
2. Pop Operation
3. Display All Elements
4. Print Top Element
5. Exit
Enter Your Choice: 1
Enter the Element to be insertd: 98
Select Your Choice:
1. Push Operation
2. Pop Operation
3. Display All Elements
4. Print Top Element
5. Exit
Enter Your Choice: 1
Enter the Element to be insertd: 74
Select Your Choice:
1. Push Operation
2. Pop Operation
3. Display All Elements
4. Print Top Element
5. Exit
Enter Your Choice: 3
74      98      32
Select Your Choice:
1. Push Operation
2. Pop Operation
3. Display All Elements
4. Print Top Element
5. Exit
Enter Your Choice: 4
Top Elements is: 74
Select Your Choice:
1. Push Operation
2. Pop Operation
3. Display All Elements
4. Print Top Element
5. Exit
Enter Your Choice:
```

Experiment 12: Program to implement push and pop operations on a queue using linked list

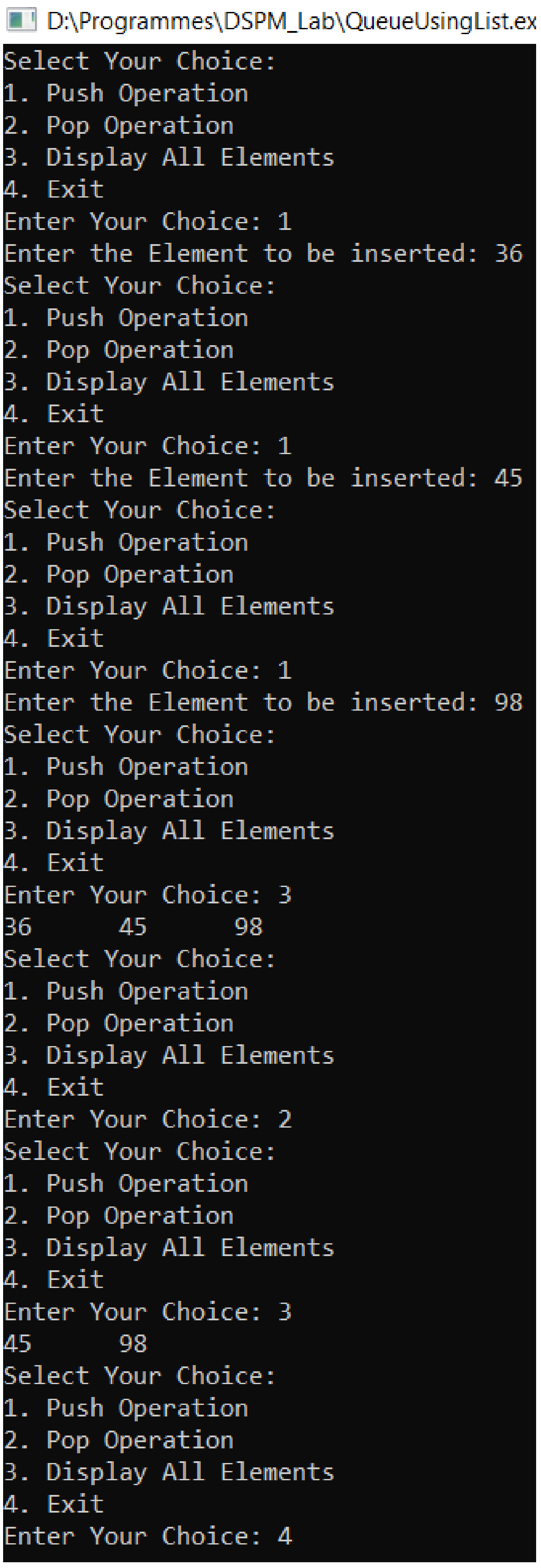
Code:

```
#include<stdlib.h>
#include<stdio.h>
struct queue{
    int data;
    struct queue* next;
};
struct queue* front = NULL;
struct queue* rear = NULL;

void enqueue(int data){
    struct queue* tempNode = (struct queue*)malloc(sizeof(struct
queue));
    tempNode->data = data;
    tempNode->next = NULL;

    if(front == NULL && rear == NULL){
        front = rear = tempNode;
        return;
    }
    rear->next = tempNode;
    rear = tempNode;
```

```
}

void dequeue(){
    struct queue* tempNode = front;
    if(front == NULL)
        return;
    if(front == rear){
        front = rear = NULL;
    }
    else{
        front = front->next;
    }

    free(tempNode);
}

void display(){
    struct queue* tempNode = front;
    while(tempNode != NULL){
        printf("%d \t",tempNode->data);
        tempNode = tempNode->next;
    }
    printf("\n");
```

```
}
int main(){
    int data,choice;
    while(1){
        printf("Select Your Choice: \n");
        printf("1. Push Operation \n");
        printf("2. Pop Operation \n");
        printf("3. Display All Elements \n");
        printf("4. Exit \n");
        printf("Enter Your Choice: ");
        scanf("%d",&choice);
        switch(choice){
            case 1: {
                printf("Enter the Element to be inserted: ");
                scanf("%d",&data);
                enqueue(data);
                break;
            }
            case 2: {
                dequeue();
                break;
            }
            case 3: {
                display();
                break;
            }
            case 4: {
                exit(0);
                break;
            }
            default: {
                printf("Invalid Choice!!\n");
                break;
            }
        }
    }

    return 0;
}
```

Output:



D:\Programmes\DSPM_Lab\QueueUsingList.ex

```
Select Your Choice:
1. Push Operation
2. Pop Operation
3. Display All Elements
4. Exit
Enter Your Choice: 1
Enter the Element to be inserted: 36
Select Your Choice:
1. Push Operation
2. Pop Operation
3. Display All Elements
4. Exit
Enter Your Choice: 1
Enter the Element to be inserted: 45
Select Your Choice:
1. Push Operation
2. Pop Operation
3. Display All Elements
4. Exit
Enter Your Choice: 1
Enter the Element to be inserted: 98
Select Your Choice:
1. Push Operation
2. Pop Operation
3. Display All Elements
4. Exit
Enter Your Choice: 3
36      45      98
Select Your Choice:
1. Push Operation
2. Pop Operation
3. Display All Elements
4. Exit
Enter Your Choice: 2
Select Your Choice:
1. Push Operation
2. Pop Operation
3. Display All Elements
4. Exit
Enter Your Choice: 3
45      98
Select Your Choice:
1. Push Operation
2. Pop Operation
3. Display All Elements
4. Exit
Enter Your Choice: 4
```

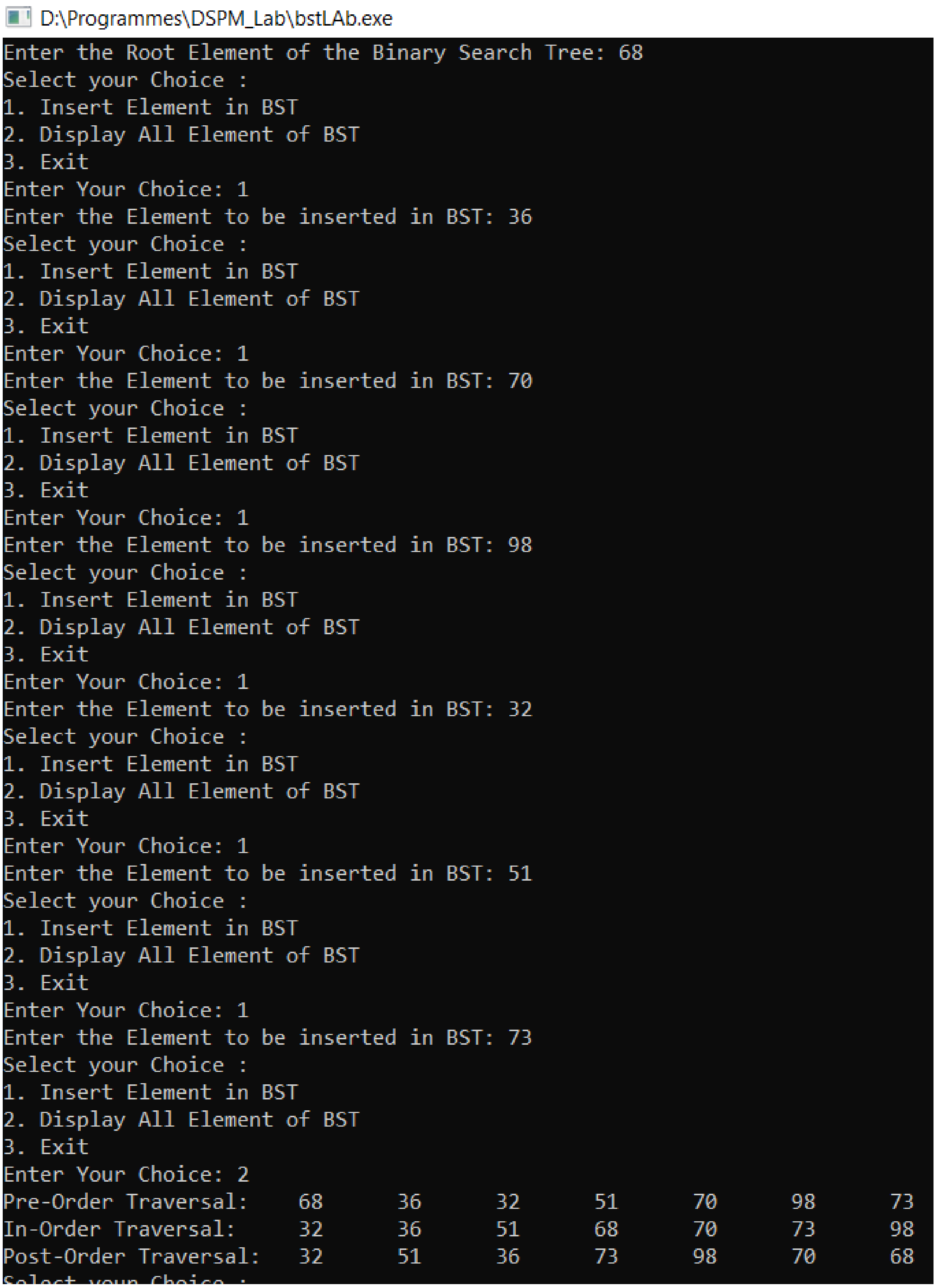
Experiment 13: Program to traverse a Binary search tree in Pre-order, In-order and Post-order

Code:

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

struct tree{
    int data;
    struct tree* leftChild;
    struct tree* rightChild;
};

struct tree* root = NULL;
int newNode(int data){
    struct tree* tempNode = (struct tree*)malloc(sizeof(struct
tree));
    tempNode->data = data;
    tempNode->leftChild = NULL;
    tempNode->rightChild = NULL;
    return tempNode;
}
int insertinBST(struct tree* tempNode,int data){

    if(tempNode == NULL){
        return newNode(data);
    }

    if(data < tempNode->data){
        tempNode->leftChild = insertinBST(tempNode-
>leftChild,data);
    }
    else{
        tempNode->rightChild = insertinBST(tempNode-
>rightChild,data);
    }

    return tempNode;
}

void preorder(struct tree* tempNode) {
   if(tempNode != NULL) {
      printf("%d \t",tempNode->data);
      preorder(tempNode->leftChild);
      preorder(tempNode->rightChild);
   }
}

void inorder(struct tree* tempNode) {
   if(tempNode != NULL) {
      inorder(tempNode->leftChild);
      printf("%d \t",tempNode->data);
      inorder(tempNode->rightChild);
   }
}

void postorder(struct tree* tempNode) {
   if(tempNode != NULL) {
      postorder(tempNode->leftChild);
      postorder(tempNode->rightChild);
      printf("%d \t", tempNode->data);
   }
}
void display(){
    printf("Pre-Order Traversal: \t");
    preorder(root);
    printf("\nIn-Order Traversal: \t");
    inorder(root);
    printf("\nPost-Order Traversal: \t");
    postorder(root);
    printf("\n");
}
```

```
int main(){

    int choice,data,result;
    printf("Enter the Root Element of the Binary Search Tree: ");
    scanf("%d",&data);
    root = insertinBST(root,data);

    while(1){

        printf("Select your Choice : \n");
        printf("1. Insert Element in BST\n");
        printf("2. Display All Element of BST\n");
        printf("3. Exit\n");
        printf("Enter Your Choice: ");
        scanf("%d",&choice);
        switch (choice)
        {
            case 1:{
                printf("Enter the Element to be inserted in BST: ");
                scanf("%d",&data);
                insertinBST(root,data);
                break;
            }
            case 2:{
                display();
                break;
            }
            case 3:{
                exit(0);
                break;
            }
            default:{
                printf("Invalid Choice!! ");
                break;
            }
        }
    }

    return 0;
}
```

Output:



D:\Programmes\DSPM_Lab\bstLAb.exe

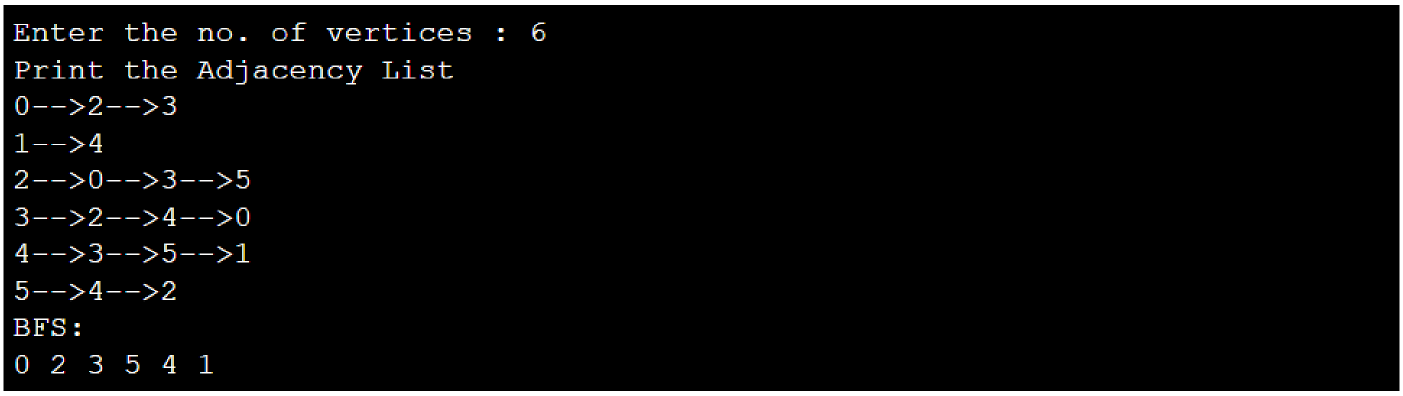
```
Enter the Root Element of the Binary Search Tree: 68
Select your Choice :
1. Insert Element in BST
2. Display All Element of BST
3. Exit
Enter Your Choice: 1
Enter the Element to be inserted in BST: 36
Select your Choice :
1. Insert Element in BST
2. Display All Element of BST
3. Exit
Enter Your Choice: 1
Enter the Element to be inserted in BST: 70
Select your Choice :
1. Insert Element in BST
2. Display All Element of BST
3. Exit
Enter Your Choice: 1
Enter the Element to be inserted in BST: 98
Select your Choice :
1. Insert Element in BST
2. Display All Element of BST
3. Exit
Enter Your Choice: 1
Enter the Element to be inserted in BST: 32
Select your Choice :
1. Insert Element in BST
2. Display All Element of BST
3. Exit
Enter Your Choice: 1
Enter the Element to be inserted in BST: 51
Select your Choice :
1. Insert Element in BST
2. Display All Element of BST
3. Exit
Enter Your Choice: 1
Enter the Element to be inserted in BST: 73
Select your Choice :
1. Insert Element in BST
2. Display All Element of BST
3. Exit
Enter Your Choice: 2
Pre-Order Traversal:    68      36      32      51      70      98      73
In-Order Traversal:     32      36      51      68      70      73      98
Post-Order Traversal:   32      51      36      73      98      70      68
```

Experiment 14: Program to traverse graphs using BFS.
Code:

```
#include <bits/stdc++.h>
using namespace std;

//	Make a pair between vertex x and vertex y
void addedge(list<int> *ls,int x,int y){
	ls[x].push_back(y);
	ls[y].push_back(x);
	return;
}

//Breath First Search of a Graph
void BFS(list<int>*ls,int num,int x){
	bool *visit= new bool[num];
	for(int i=0;i<num;i++){
		visit[i]=false;
	}
	queue<int> q;
	q.push(x);
	while(!q.empty()){
		int s=q.front();
		q.pop();
		if(!visit[s]){
			visit[s]=true;
			cout<<s<<" ";
			list<int>::iterator it;
			for(it=ls[s].begin();it!=ls[s].end();it++){
				q.push(*it);
			}
		}
	}
}
```

```
// Print the Adjacency List
void print(list<int> *ls,int num){
	list<int>::iterator it;
	for(int i=0;i<6;i++){
		cout<<i<<"-->";
		for(it=ls[i].begin();it!=ls[i].end();it++){
			cout<<*it<<"-->";
		}
		cout<<endl;
	}
}

int main(){
	int num=6;

	cout<<"Enter the no. of vertices : 6\n";
	list<int> *ls=new list<int>[num];

	addedge(ls,0,2);
	addedge(ls,2,3);
	addedge(ls,3,4);
	addedge(ls,4,5);
	addedge(ls,2,5);
	addedge(ls,1,4);
	addedge(ls,3,0);
	cout<<"Print of adjacency list:"<<endl;
	print(ls,6);
	cout<<"BFS"<<endl;
	BFS(ls,6,0);

	return 0;
}
```

Output:



```
Enter the no. of vertices : 6
Print the Adjacency List
0-->2-->3
1-->4
2-->0-->3-->5
3-->2-->4-->0
4-->3-->5-->1
5-->4-->2
BFS:
0 2 3 5 4 1
```

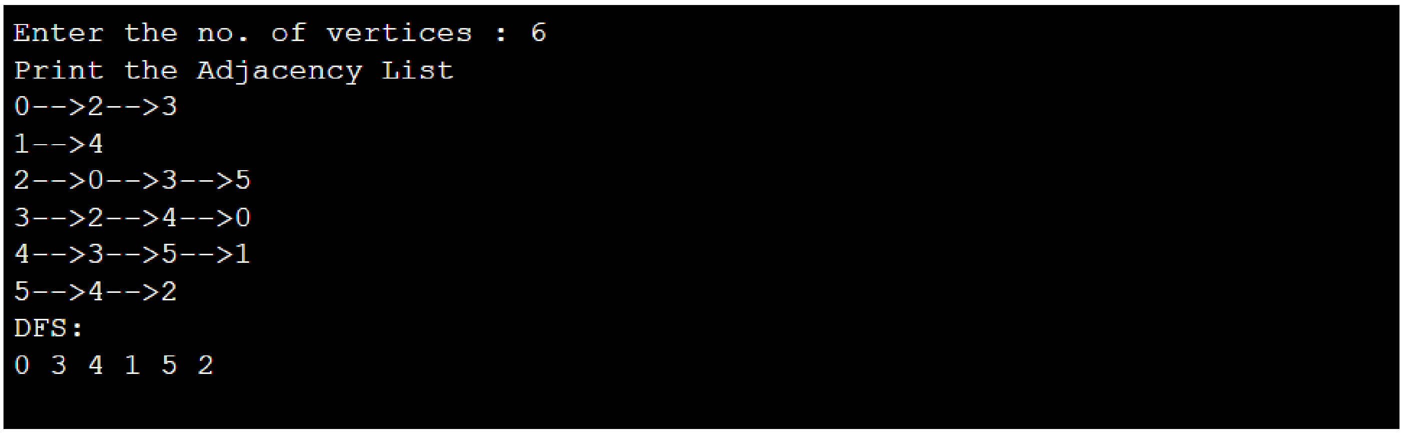
Experiment 15: Program to traverse graphs using DFS.

Code:

```
#include <iostream>
#include <bits/stdc++.h>
#include <list>
using namespace std;

void addedge(list<int>,int ,int );
void DFS(list<int>*,int);

//      Make a pair between vertex x and vertex y
void addedge(list<int> *ls,int x,int y){
        ls[x].push_back(y);
        ls[y].push_back(x);
        return;
}

//Depth First Search of a Graph
void DFS(list<int>*ls,int num,int x){
        bool *visit= new bool[num];
        for(int i=0;i<num;i++){
                visit[i]=false;
        }
        stack<int> st;
        st.push(x);
        while(!st.empty()){
                int s=st.top();
                st.pop();
                if(!visit[s]){
                        cout<<s<< " ";
                        visit[s]=true;
                        list<int>::iterator it;
                        for(it=ls[s].begin();it!=ls[s].end();it++){
```

```
                        st.push(*it);
                    }
                }
            }
}

// Print the Adjacency List
void print(list<int> *ls,int num){
        list<int>::iterator it;
        for(int i=0;i<6;i++){
                cout<<i<<"-->";
                for(it=ls[i].begin();it!=ls[i].end();it++){
                        cout<<*it<<"-->";
                }
                cout<<endl;
        }
}

int main(){
        int num=6;

        cout<<"Enter the no. of vertices : 6\n";
        list<int> *ls=new list<int>[num];

        addedge(ls,0,2);
        addedge(ls,2,3);
        addedge(ls,3,4);
        addedge(ls,4,5);
        addedge(ls,2,5);
        addedge(ls,1,4);
        addedge(ls,3,0);
        cout<<"Print of adjacency matrix:"<<endl;
        print(ls,6);
        cout<<"DFS"<<endl;
        DFS(ls,6,0);

        return 0;
}
```

Output:



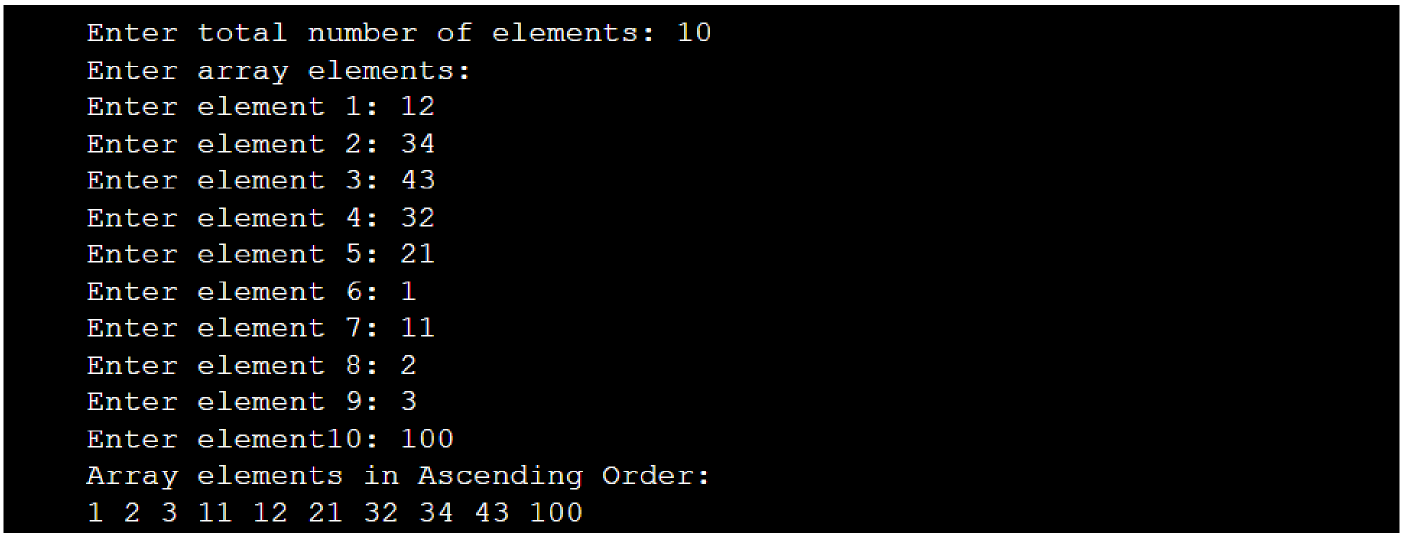
```
Enter the no. of vertices : 6
Print the Adjacency List
0-->2-->3
1-->4
2-->0-->3-->5
3-->2-->4-->0
4-->3-->5-->1
5-->4-->2
DFS:
0 3 4 1 5 2
```

Experiment 16: Program to sort an array of integers in ascending order using bubble sort.

Code:

```
#include <stdio.h>
#define MAX 100
int main()
{
    int arr[MAX],limit;
    int i,j,temp;

    printf("Enter total number of elements: ");
    scanf("%d",&limit);

    /*Read array*/
    printf("Enter array elements: \n");
    for(i=0; i<limit; i++)
    {
        printf("Enter element %3d: ",i+1);
        scanf("%d",&arr[i]);
    }

    /*sort elements in Ascending Order*/
    for(i=0; i<(limit-1); i++)
    {
        for(j=0; j<(limit-i-1); j++)
        {
            if(arr[j]>arr[j+1])
            {
                temp=arr[j];
                arr[j]=arr[j+1];
                arr[j+1]=temp;
            }
        }
    }

    printf("Array elements in Ascending Order:\n");
    for(i=0; i<limit; i++)
        printf("%d ",arr[i]);

    printf("\n");

return 0;
}
```

Output:



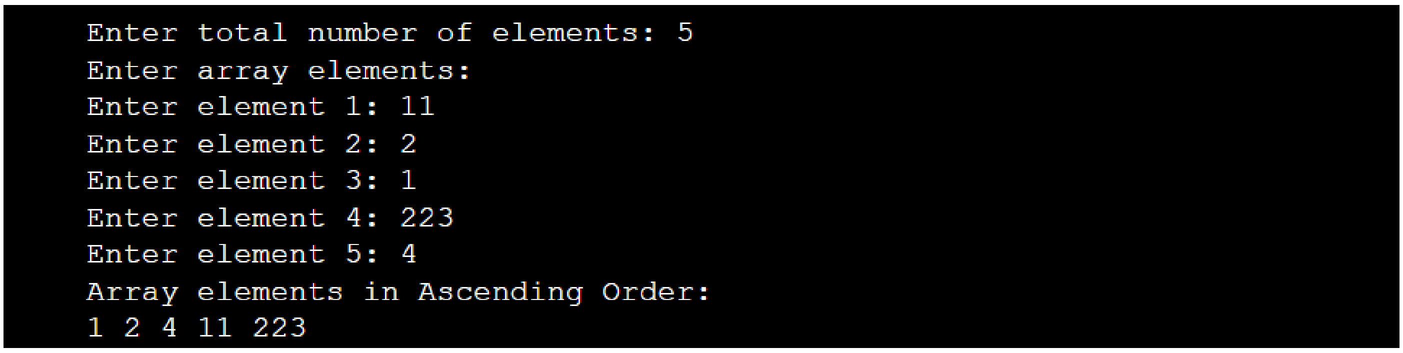
```
Enter total number of elements: 10
Enter array elements:
Enter element 1: 12
Enter element 2: 34
Enter element 3: 43
Enter element 4: 32
Enter element 5: 21
Enter element 6: 1
Enter element 7: 11
Enter element 8: 2
Enter element 9: 3
Enter element10: 100
Array elements in Ascending Order:
1 2 3 11 12 21 32 34 43 100
```

Experiment 17: Program to sort an array of integers in ascending order using selection sort.

Code:

```
#include <stdio.h>

#define MAX 100

int main()
{
    int arr[MAX],limit;
    int i,j,temp,position;

    printf("Enter total number of elements: ");
    scanf("%d",&limit);

    /*Read array*/
    printf("Enter array elements: \n");
    for(i=0; i<limit; i++)
    {
        printf("Enter element %3d: ",i+1);
        scanf("%d",&arr[i]);
    }

    /*sort elements in Ascending Order*/
    for(i=0; i<(limit); i++)
    {
        position=i;
        for(j=i+1; j<limit; j++)
        {
            if(arr[position]>arr[j])
            {
                position=j;
            }
            if(position!=i)
            {
                temp=arr[i];
                arr[i]=arr[position];
                arr[position]=temp;
            }
        }
    }

    printf("Array elements in Ascending Order:\n");
    for(i=0; i<limit; i++)
        printf("%d ",arr[i]);

    printf("\n");
return 0;
}
```

Output:



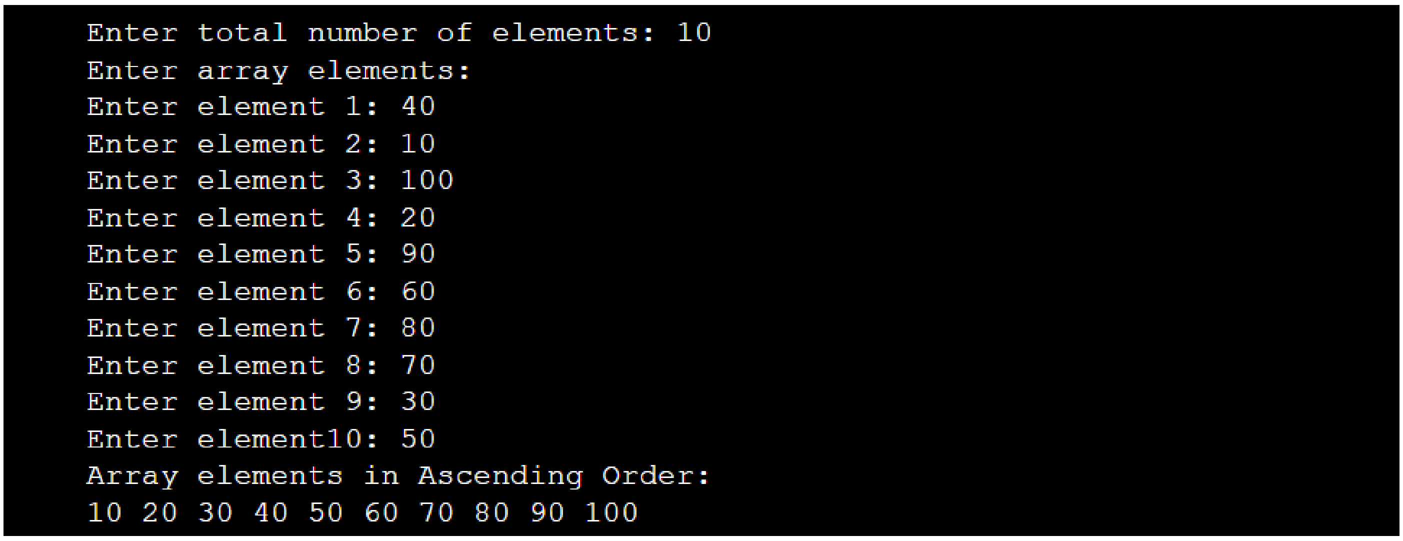
```
Enter total number of elements: 5
Enter array elements:
Enter element 1: 11
Enter element 2: 2
Enter element 3: 1
Enter element 4: 223
Enter element 5: 4
Array elements in Ascending Order:
1 2 4 11 223
```

Experiment 18: Program to sort an array of integers in ascending order using insertion sort.

Code:

```
#include <stdio.h>

#define MAX 100

int main()
{
    int arr[MAX],limit;
    int i,j,temp;

    printf("Enter total number of elements: ");
    scanf("%d",&limit);

    /*Read array*/
    printf("Enter array elements: \n");
    for(i=0; i<limit; i++)
    {
        printf("Enter element %3d: ",i+1);
        scanf("%d",&arr[i]);
    }

    /*sort elements in Ascending Order*/
    for(i=1; i<(limit); i++)
    {
        j=i;
        while(j>0 && arr[j]<arr[j-1])
        {
            temp=arr[j];
            arr[j]=arr[j-1];
            arr[j-1]=temp;
            j--;
         }
      }

      printf("Array elements in Ascending Order:\n");
      for(i=0; i<limit; i++)
         printf("%d ",arr[i]);

      printf("\n");
   return 0;
   }
```

Output:



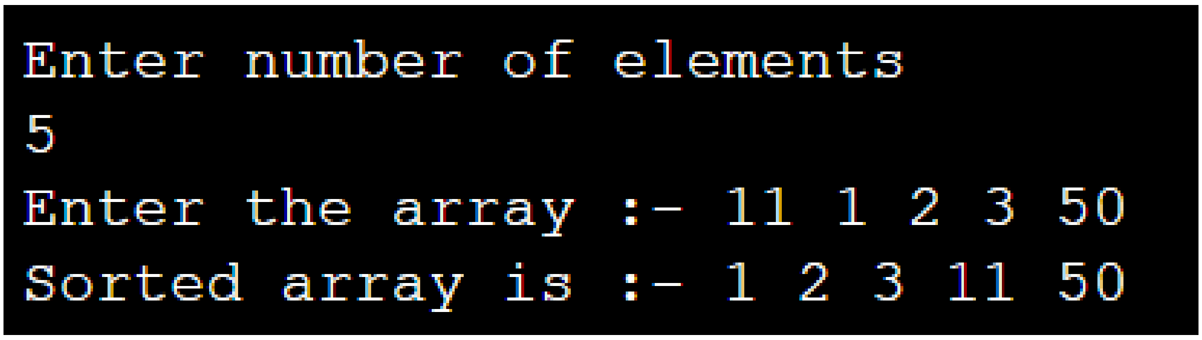
```
Enter total number of elements: 10
Enter array elements:
Enter element 1: 40
Enter element 2: 10
Enter element 3: 100
Enter element 4: 20
Enter element 5: 90
Enter element 6: 60
Enter element 7: 80
Enter element 8: 70
Enter element 9: 30
Enter element10: 50
Array elements in Ascending Order:
10 20 30 40 50 60 70 80 90 100
```

Experiment 18: Program to sort an array of integers in ascending order using quick sort.

Code:

```
#include<iostream>
using namespace std;

void quicksort(int ,int ,int);
int partition(int ,int,int);
int partition(int *a,int s,int e)
{
        int piviot=a[e];
    int pind=s;
    int i,t;
    for(i=s;i<e;i++)
    {
        if(a[i]<=piviot)
        {
        t=a[i];
            a[i]=a[pind];
            a[pind]=t;
            pind++;
        }
        }

    t=a[e];
    a[e]=a[pind];
    a[pind]=t;
    return pind;
}
void quicksort(int *a,int s,int e)
{
        if(s<e)
        {
            int pind=partition(a,s,e);
        quicksort(a,s,pind-1);
        quicksort(a,pind+1,e);
        }
}
int main()
{
        int n;
    cout<<"Enter number of elements"<<endl;
    cin>>n;
    int a[n];
    cout<<"Enter the array :- ";
    for(int i=0;i<n;i++)
    {
        cin>>a[i];
        }
        quicksort(a,0,n-1);
    cout<<"Sorted array is :- ";
    for(int i=0;i<n;i++)
    {
        cout<<a[i]<<" ";
        }
        return 0;
}
```

Output:



```
Enter number of elements
5
Enter the array :- 11 1 2 3 50
Sorted array is :- 1 2 3 11 50
```