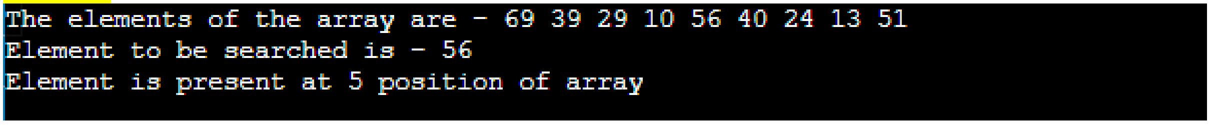
Data Structure

-: Lab Manual Solution :-

1. Write a program to implement linear search to find an item in the list.

```
#include <iostream>
using namespace std;
int linearSearch(int a[ ], int n, int val) {
  // Going through array linearly
  for (int i = 0; i < n; i++)
    {
        if (a[i] == val)
        return i+1;
    }
  return -1;
}
int main() {
  int a[ ] = {69, 39, 29, 10, 56, 40, 24, 13, 51}; // given array
  int val = 56; // value to be searched
  int n = sizeof(a) / sizeof(a[0]); // size of array
  int res = linearSearch(a, n, val); // Store result
  cout<<"The elements of the array are - ";
  for (int i = 0; i < n; i++)
  cout<<a[i]<<" ";
  cout<<"\nElement to be searched is - "<<val;
  if (res == -1)
  cout<<"\nElement is not present in the array";
  else
  cout<<"\nElement is present at "<<res<<" position of array";
  return 0;
}
```

OutPut-:



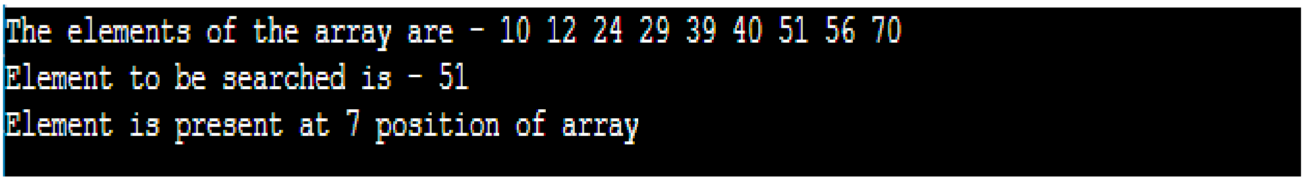
```
The elements of the array are - 69 39 29 10 56 40 24 13 51
Element to be searched is - 56
Element is present at 5 position of array
```

2. Write a program to implement binary search to find an element in an ordered list.

```
#include <iostream>
using namespace std;
int binarySearch(int a[ ], int beg, int end, int val)
{
    int mid;
    if(end >= beg)
    {
        mid = (beg + end)/2;
/* if the item to be searched is present at middle */
        if(a[mid] == val)
        {
            return mid+1;
        }
            /* if the item to be searched is smaller than middle, then it can only be in left sub array */
        else if(a[mid] < val)
        {
            return binarySearch(a, mid+1, end, val);
        }
        /* if the item to be searched is greater than middle, then it can only be in right subarray */
    else
        {
            return binarySearch(a, beg, mid-1, val);
        }
    }
    return -1;
}
int main() {
  int a[ ] = {10, 12, 24, 29, 39, 40, 51, 56, 70}; // given array
  int val = 51; // value to be searched
  int n = sizeof(a) / sizeof(a[0]); // size of array
  int res = binarySearch(a, 0, n-1, val); // Store result
  cout<<"The elements of the array are - ";
  for (int i = 0; i < n; i++)
```

```
 cout<<a[i]<<" ";
 cout<<"\nElement to be searched is - "<<val;
 if (res == -1)
 cout<<"\nElement is not present in the array";
 else
 cout<<"\nElement is present at "<<res<<" position of array";
 return 0;
}
```

OutPut:-



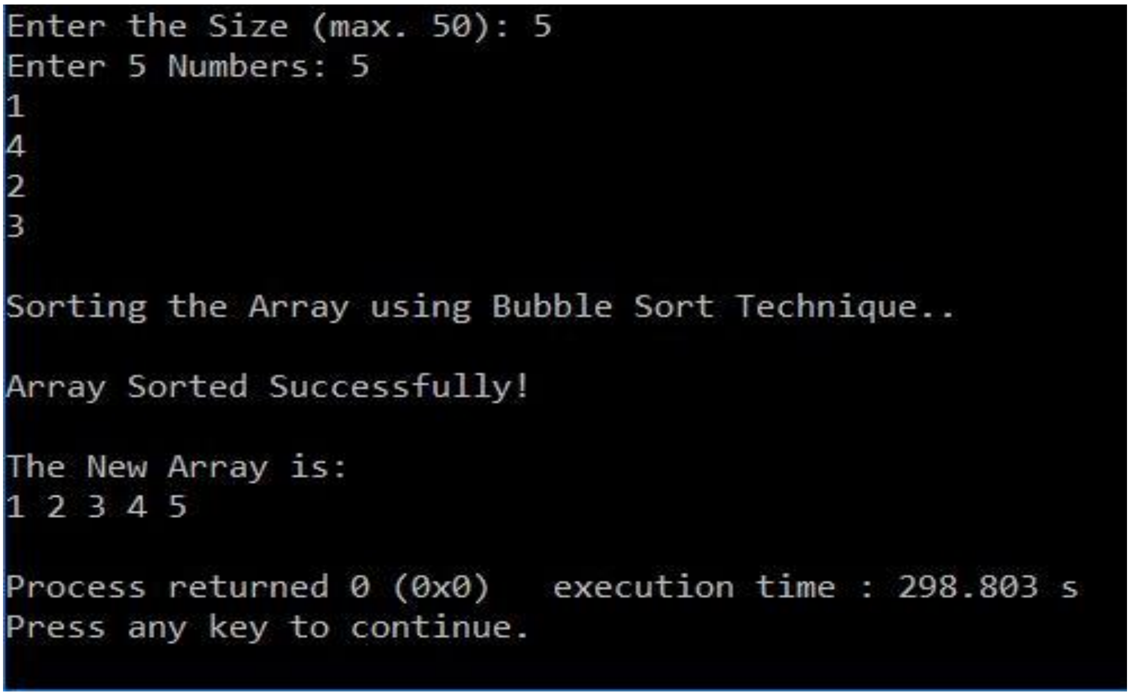
```
The elements of the array are - 10 12 24 29 39 40 51 56 70
Element to be searched is - 51
Element is present at 7 position of array
```

3. Write a program to sort given elements using a bubble sort algorithm.

```
#include<iostream>
using namespace std;
int main()
{
   int n, i, arr[50], j, temp;
   cout<<"Enter the Size (max. 50): ";
   cin>>n;
   cout<<"Enter "<<n<<" Numbers: ";
   for(i=0; i<n; i++)
      cin>>arr[i];
   cout<<"\nSorting the Array using Bubble Sort Technique..\n";
   for(i=0; i<(n-1); i++)
   {
      for(j=0; j<(n-i-1); j++)
      {
         if(arr[j]>arr[j+1])
         {
            temp = arr[j];
            arr[j] = arr[j+1];
```

```
                arr[j+1] = temp;
            }
        }
    }
    cout<<"\nArray Sorted Successfully!\n";
    cout<<"\nThe New Array is: \n";
    for(i=0; i<n; i++)
        cout<<arr[i]<<" ";
    cout<<endl;
    return 0;
}
```

OutPut:-



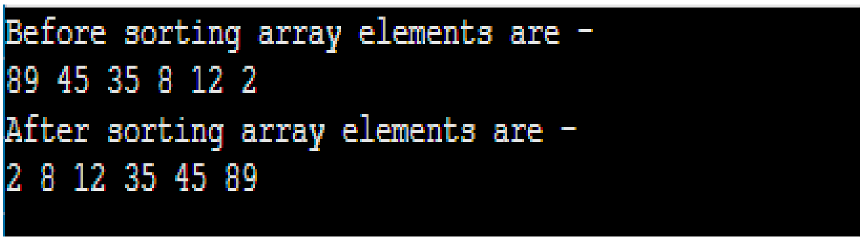
```
Enter the Size (max. 50): 5
Enter 5 Numbers: 5
1
4
2
3

Sorting the Array using Bubble Sort Technique..

Array Sorted Successfully!

The New Array is:
1 2 3 4 5

Process returned 0 (0x0)   execution time : 298.803 s
Press any key to continue.
```

4. Write a program to sort given elements using a insertion sort algorithm.

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

void insert(int a[ ], int n) /* function to sort an aay with insertion sort */
{
   int i, j, temp;
   for (i = 1; i < n; i++) {
      temp = a[i];
      j = i - 1;

      while(j>=0 && temp <= a[j])  /* Move the elements greater than temp to one position ahead from their current position*/
      {
         a[j+1] = a[j];
         j = j-1;
      }
      a[j+1] = temp;
   }
}

void printArr(int a[ ], int n) /* function to print the array */
{
   int i;
   for (i = 0; i < n; i++)
      cout << a[i] <<" ";
}

int main()
{
   int a[ ] = { 89, 45, 35, 8, 12, 2 };
   int n = sizeof(a) / sizeof(a[0]);
   cout<<"Before sorting array elements are - "<<endl;
   printArr(a, n);
   insert(a, n);
   cout<<"\nAfter sorting array elements are - "<<endl;
```

```
    printArr(a, n);

    return 0;
}
```

OutPut:-



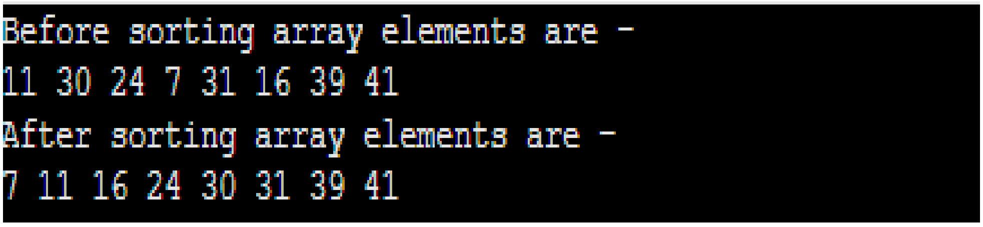
```
Before sorting array elements are -
89 45 35 8 12 2
After sorting array elements are -
2 8 12 35 45 89
```

5. Write a program to sort given elements using a merge sort algorithm.

```
#include <iostream>

using namespace std;

/* Function to merge the subarrays of a[] */
void merge(int a[], int beg, int mid, int end)
{
    int i, j, k;
    int n1 = mid - beg + 1;
    int n2 = end - mid;

    int LeftArray[n1], RightArray[n2]; //temporary arrays

    /* copy data to temp arrays */
    for (int i = 0; i < n1; i++)
    LeftArray[i] = a[beg + i];
    for (int j = 0; j < n2; j++)
    RightArray[j] = a[mid + 1 + j];

    i = 0; /* initial index of first sub-array */
    j = 0; /* initial index of second sub-array */
    k = beg;  /* initial index of merged sub-array */

    while (i < n1 && j < n2)
    {
            if(LeftArray[i] <= RightArray[j])
            {
                a[k] = LeftArray[i];
                i++;
            }
            else
            {
                a[k] = RightArray[j];
                j++;
            }
            k++;
        }
        while (i<n1)
        {
            a[k] = LeftArray[i];
            i++;
            k++;
        }

        while (j<n2)
        {
            a[k] = RightArray[j];
            j++;
            k++;
        }
    }

    void mergeSort(int a[], int beg, int end)
    {
        if (beg < end)
        {
            int mid = (beg + end) / 2;
            mergeSort(a, beg, mid);
            mergeSort(a, mid + 1, end);
            merge(a, beg, mid, end);
        }
    }
```

```
/* Function to print the array */
void printArray(int a[], int n)
{
    int i;
    for (i = 0; i < n; i++)
        cout<<a[i]<<" ";
}

int main()
{
    int a[] = { 11, 30, 24, 7, 31, 16, 39, 41 };
    int n = sizeof(a) / sizeof(a[0]);
    cout<<"Before sorting array elements are - \n";
    printArray(a, n);
    mergeSort(a, 0, n - 1);
    cout<<"\nAfter sorting array elements are - \n";
    printArray(a, n);
    return 0;
}
```

OutPut;-



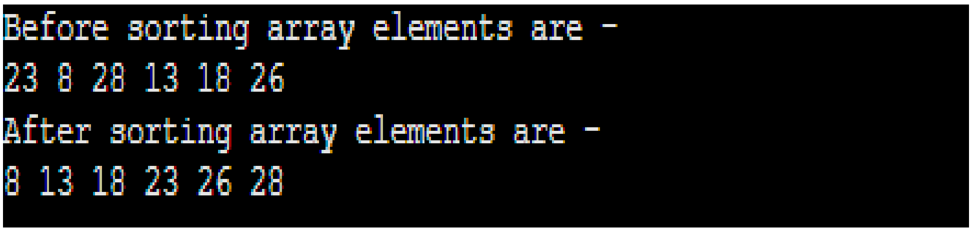
```
Before sorting array elements are -
11 30 24 7 31 16 39 41
After sorting array elements are -
7 11 16 24 30 31 39 41
```

6. Write a program to sort given elements using a quick sort Algorithm.

```
#include <iostream>

using namespace std;

/* function that consider last element as pivot,
place the pivot at its exact position, and place
smaller elements to left of pivot and greater
elements to right of pivot.  */
int partition (int a[ ], int start, int end)
{
    int pivot = a[end]; // pivot element
    int i = (start - 1);

    for (int j = start; j <= end - 1; j++)
    {
        // If current element is smaller than the pivot
        if (a[j] < pivot)
        {
            i++; // increment index of smaller element
            int t = a[i];
            a[i] = a[j];
            a[j] = t;
        }
    }
    int t = a[i+1];
    a[i+1] = a[end];
    a[end] = t;
    return (i + 1);
}

/* function to implement quick sort */
void quick(int a[ ], int start, int end) /* a[] = array to be sorted,
start = Starting index, end = Ending index */
{
    if (start < end)
    {
        int p = partition(a, start, end);  //p is the partitioning index
        quick(a, start, p - 1);
        quick(a, p + 1, end);
    }
}

/* function to print an array */
void printArr(int a[], int n)
{
    int i;
    for (i = 0; i < n; i++)
        cout<<a[i]<< " ";
```

```
}
int main()
{
    int a[] = { 23, 8, 28, 13, 18, 26 };
    int n = sizeof(a) / sizeof(a[0]);
    cout<<"Before sorting array elements are - \n";
    printArr(a, n);
    quick(a, 0, n - 1);
    cout<<"\nAfter sorting array elements are - \n";
    printArr(a, n);

    return 0;
}
```

OutPut-:

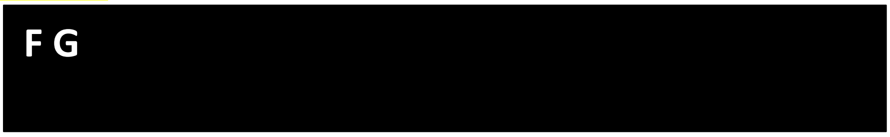


```
Before sorting array elements are -
23 8 28 13 18 26
After sorting array elements are -
8 13 18 23 26 28
```

7. Write a program to implement various set operations on a given set of elements.

```
#include <iostream>
#include <set>

int main()
{
    std::set<char> a;
    a.insert('G');
    a.insert('F');
    a.insert('G');
    for (auto& str : a) {
        std::cout << str << ' ';
    }
    std::cout << '\n';
    return 0;
}
```

OutPut

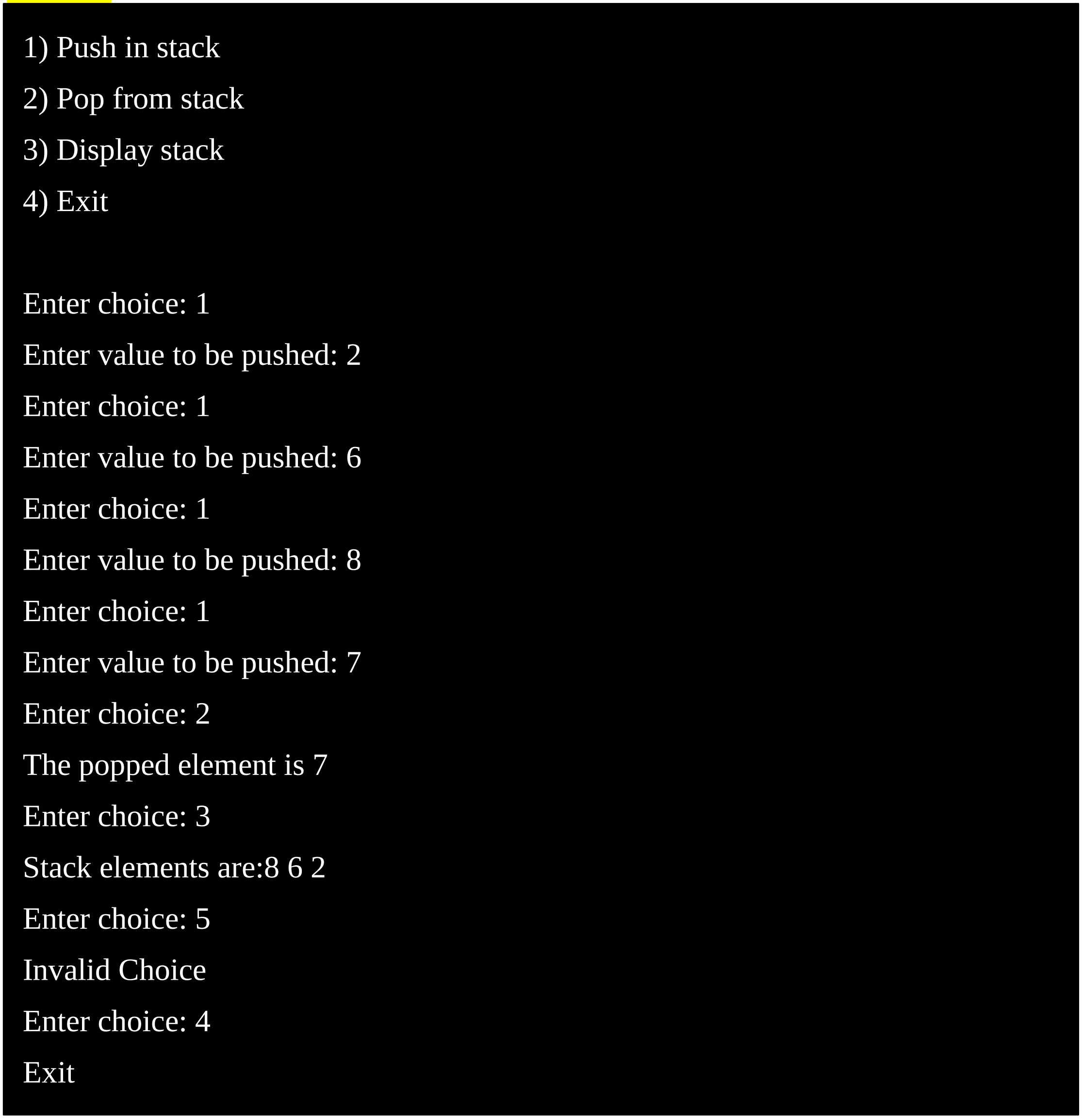


```
F G
```

8. Write a program to implement a stack and perform various operations like push, pop, display.

```
#include <iostream>
using namespace std;
int stack[100], n=100, top=-1;
void push(int val) {
  if(top>=n-1)
  cout<<"Stack Overflow"<<endl;
  else {
    top++;
    stack[top]=val;
  }
}
void pop() {
  if(top<=-1)
  cout<<"Stack Underflow"<<endl;
  else {
    cout<<"The popped element is "<< stack[top] <<endl;
    top--;
  }
}
void display() {
  if(top>=0) {
    cout<<"Stack elements are:";
    for(int i=top; i>=0; i--)
    cout<<stack[i]<<" ";
    cout<<endl;
  } else
  cout<<"Stack is empty";
}
int main() {
   int ch, val;
   cout<<"1) Push in stack"<<endl;
   cout<<"2) Pop from stack"<<endl;
   cout<<"3) Display stack"<<endl;
   cout<<"4) Exit"<<endl;
   do {
      cout<<"Enter choice: "<<endl;
      cin>>ch;
      switch(ch) {
         case 1: {
            cout<<"Enter value to be pushed:"<<endl;
            cin>>val;
            push(val);
            break;
         }
         case 2: {
            pop();
            break;
         }
         case 3: {
            display();
            break;
         }
         case 4: {
            cout<<"Exit"<<endl;
            break;
         }
         default: {
            cout<<"Invalid Choice"<<endl;
         }
      }
   }while(ch!=4);
   return 0;
}
```

OutPut:-



```
1) Push in stack
2) Pop from stack
3) Display stack
4) Exit

Enter choice: 1
Enter value to be pushed: 2
Enter choice: 1
Enter value to be pushed: 6
Enter choice: 1
Enter value to be pushed: 8
Enter choice: 1
Enter value to be pushed: 7
Enter choice: 2
The popped element is 7
Enter choice: 3
Stack elements are:8 6 2
Enter choice: 5
Invalid Choice
Enter choice: 4
Exit
```

9. Write a program to implement a queue and perform various operations on the queue.

```
#include <iostream>
using namespace std;
int queue[100], n = 100, front = - 1, rear = - 1;
void Insert() {
  int val;
  if (rear == n - 1)
  cout<<"Queue Overflow"<<endl;
  else {
      if (front == - 1)
      front = 0;
      cout<<"Insert the element in queue : "<<endl;
      cin>>val;
      rear++;
      queue[rear] = val;
   }
}
void Delete() {
   if (front == - 1 || front > rear) {
      cout<<"Queue Underflow ";
      return ;
   } else {
      cout<<"Element deleted from queue is : "<< queue[front] <<endl;
      front++;;
   }
}
void Display() {
   if (front == - 1)
   cout<<"Queue is empty"<<endl;
   else {
      cout<<"Queue elements are : ";
      for (int i = front; i <= rear; i++)
      cout<<queue[i]<<" ";
         cout<<endl;
   }
}
int main() {
   int ch;
   cout<<"1) Insert element to queue"<<endl;
   cout<<"2) Delete element from queue"<<endl;
   cout<<"3) Display all the elements of queue"<<endl;
   cout<<"4) Exit"<<endl;
   do {
      cout<<"Enter your choice : "<<endl;
      cin>>ch;
      switch (ch) {
         case 1: Insert();
         break;
         case 2: Delete();
         break;
         case 3: Display();
        break;
         case 4: cout<<"Exit"<<endl;
         break;
         default: cout<<"Invalid choice"<<endl;
       }
   } while(ch!=4);
   return 0;
}
```

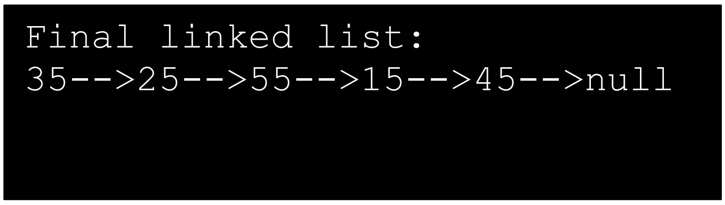
OutPut:-

```
1) Insert element to queue
2) Delete element from queue
3) Display all the elements of queue
4) Exit
Enter your choice : 1
Insert the element in queue : 4
Enter your choice : 1
Insert the element in queue : 3
Enter your choice : 1
Insert the element in queue : 5
Enter your choice : 2
Element deleted from queue is : 4
Enter your choice : 3
Queue elements are : 3 5
Enter your choice : 7
Invalid choice
Enter your choice : 4
Exit
```

10. Write a program to implement a linked list and perform various operations on the linked list.

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

struct Node
{
  int data;
  struct Node *next;
};
void push ( struct Node** head, int nodeData )
{
  struct Node* newNode1 = new Node;

  newNode1 -> data = nodeData;
  newNode1 -> next = (*head);

  (*head) = newNode1;
}
void insertAfter ( struct Node* prevNode, int nodeData )
{
if ( prevNode == NULL )
{
  cout << "the given previous node is required,cannot be NULL";
  return;

}
  struct Node* newNode1 =new Node;
  newNode1 -> data = nodeData;
  newNode1 -> next = prevNode -> next;
   prevNode -> next = newNode1;
}
void append ( struct Node** head, int nodeData )
{
struct Node* newNode1 = new Node;

struct Node *last = *head;
newNode1 -> data = nodeData;
newNode1 -> next = NULL;
if ( *head == NULL )
{
*head = newNode1;
return;
}
while ( last -> next != NULL )
last = last -> next;
last -> next = newNode1;
return;
}
void displayList ( struct Node *node )
{
  while ( node != NULL )
  {
    cout << node -> data << "-->";
    node = node -> next;
  }

if ( node== NULL)
cout<<"null";
}
int main ()
{
struct Node* head = NULL;
append ( &head, 15 );
push ( &head, 25 );
push ( &head, 35 );
append ( &head, 45 );
insertAfter ( head -> next, 55 );

cout << "Final linked list: " << endl;
displayList (head);

return 0;
}
```

OutPut:-

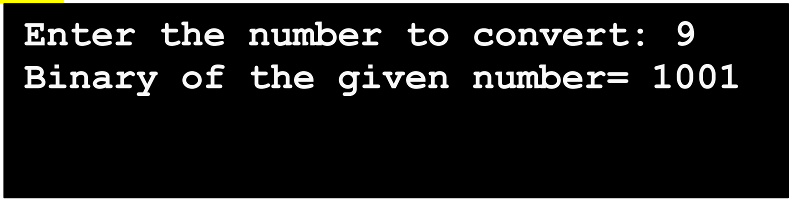


```
Final linked list:
35-->25-->55-->15-->45-->null
```

11. Write a program to convert a Decimal Number to Binary Number using Stacks.

```
#include <iostream>
using namespace std;
int main()
{
int a[10], n, i;
cout<<"Enter the number to convert: ";
cin>>n;
for(i=0; n>0; i++)
{
a[i]=n%2;
n= n/2;
}
cout<<"Binary of the given number= ";
for(i=i-1 ;i>=0 ;i--)
{
cout<<a[i];
}
}
```

OutPut-:

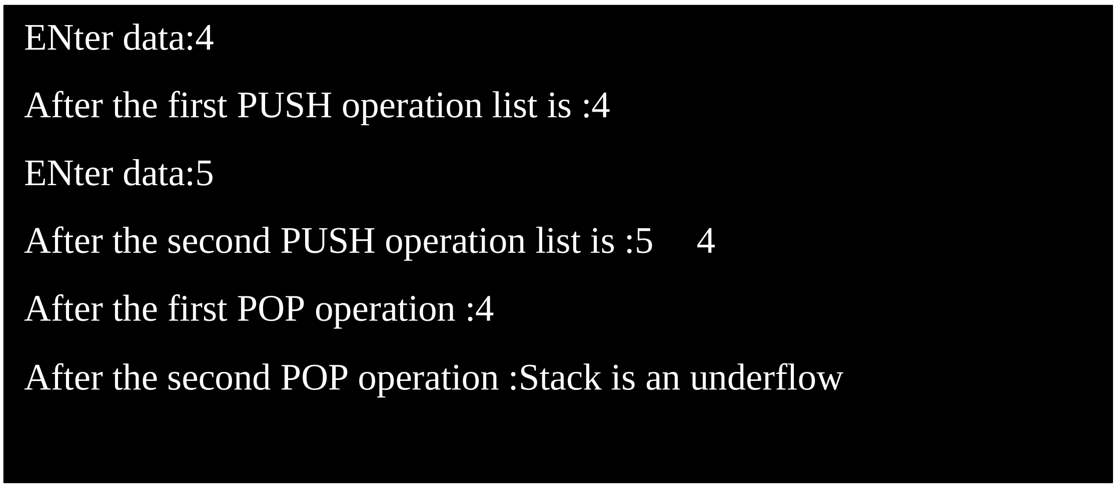


```
Enter the number to convert: 9
Binary of the given number= 1001
```

12. Write a program of STACK implementation using Linked List.

```
//stack implementation using linked list
#include<iostream>
#include<malloc.h>
using namespace std;
struct Node{
  int data;
  struct Node *next;

};

struct Node *head=NULL,*tail=NULL;

void push()
{
  struct Node *newNode;
  newNode=(struct Node*)malloc(sizeof(struct Node));
  cout<<"ENter data:";
  cin>>newNode->data;
  newNode->next=head;

  if(head==NULL)
  {
      head=tail=newNode;
  }
  else
  {
    newNode->next=head;
    head=newNode;
  }
}

void pop()
{
  struct Node *temp;
  if(head==NULL)
  {
    head=tail=NULL;
  }
  else
  {
    temp=head;
    head=head->next;
    free(temp);
  }
}

void display()
{
  struct Node *temp;
  if(head==NULL)
  {
    cout<<"Stack is an underflow";
    return;

  }
  else
  {
    temp=head;
    while(temp!=NULL)
    {
      cout<<temp->data<<"\t";
      temp=temp->next;
    }
  }
}
int main()
{
   push();
   cout<<"After the first PUSH operation list is :";
   display();
   cout<<"\n";
   push();
   cout<<"After the second PUSH operation list is :";
   display();
   cout<<"\n";
   pop();
   cout<<"After the first POP operation :";
   display();
   cout<<"\n";
    pop();
   cout<<"After the second POP operation :";
   display();
return 0;
}
```

Output-:



```
ENter data:4
After the first PUSH operation list is :4
ENter data:5
After the second PUSH operation list is :5     4
After the first POP operation :4
After the second POP operation :Stack is an underflow
```

13. Write a program to construct a binary tree, and perform different traversal operations on the same.

```
#include<iostream>
using namespace std;
struct node {
  int data;
  struct node *left;
  struct node *right;
};
struct node *createNode(int val) {
  struct node *temp = (struct node *)malloc(sizeof(struct node));
  temp->data = val;
  temp->left = temp->right = NULL;
  return temp;
}
void inorder(struct node *root) {
  if (root != NULL) {
    inorder(root->left);
    cout<<root->data<<" ";
    inorder(root->right);
  }
}
struct node* insertNode(struct node* node, int val) {
  if (node == NULL) return createNode(val);
  if (val < node->data)
  node->left = insertNode(node->left, val);
  else if (val > node->data)
  node->right = insertNode(node->right, val);
  return node;
}
int main() {
  struct node *root = NULL;
  root = insertNode(root, 4);
  insertNode(root, 5);
  insertNode(root, 2);
  insertNode(root, 9);
  insertNode(root, 1);
  insertNode(root, 3);
  cout<<"In-Order traversal of the Binary Search Tree is: ";
  inorder(root);
  return 0;
}
```

OutPut

```
In-Order traversal of the Binary Search Tree is: 1 2 3 4 5 9
```

14. Write recursive programs to implement factorial, Fibonacci series or Tower of Hanoi.

```
#include <iostream>
using namespace std;
int fib(int x) {
  if((x==1)||(x==0)) {
    return(x);
  }else {
    return(fib(x-1)+fib(x-2));
  }
}
int main() {
  int x , i=0;
  cout << "Enter the number of terms of series : ";
  cin >> x;
  cout << "\nFibonnaci Series : ";
  while(i < x) {
    cout << " " << fib(i);
    i++;
  }
  return 0;
}
```

OutPut:-

```
Enter the number of terms of series : 15
Fibonnaci Series : 0 1 1 2 3 5 8 13 21 34 55 89 144 233 377
```

15. Implement Prim’s and Kruskal Algorithm.

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

 // number of vertices in graph
 #define V 7

 // create a 2d array of size 7x7
 //for adjacency matrix to represent graph

 int main () {
   // create a 2d array of size 7x7
 //for adjacency matrix to represent graph
   int G[V][V] = {
   {0,28,0,0,0,10,0},
 {28,0,16,0,0,0,14},
 {0,16,0,12,0,0,0},
 {0,0,12,22,0,18},
 {0,0,0,22,0,25,24},
 {10,0,0,0,25,0,0},
 {0,14,0,18,24,0,0}
 };

   int edge;          // number of edge

   // create an array to check visited vertex
   int visit[V];

   //initialise the visit array to false
  for(int i=0;i<V;i++){
   visit[i]=false;
}

  // set number of edge to 0
   edge = 0;

   // the number of edges in minimum spanning tree will be
   // always less than (V -1), where V is the number of vertices in
   //graph

   // choose 0th vertex and make it true
   visit[0] = true;
```

```
int x;           //  row number
int y;           //  col number

// print for edge and weight
cout << "Edge" << " : " << "Weight";
cout << endl;
while (edge < V - 1) {//in spanning tree consist the V-1 number of edges

//For every vertex in the set S, find the all adjacent vertices
// , calculate the distance from the vertex selected.
// if the vertex is already visited, discard it otherwise
//choose another vertex nearest to selected vertex.

  int min = INT_MAX;
  x = 0;
  y = 0;

  for (int i = 0; i < V; i++) {
   if (visit[i]) {
     for (int j = 0; j < V; j++) {
      if (!visit[j] && G[i][j]) { // not in selected and there is an edge
        if (min > G[i][j]) {
          min = G[i][j];
          x = i;
          y = j;
```

```
                  }


              }
          }
        }
      }
      cout << x <<  " ---> " << y << " :  " << G[x][y];
      cout << endl;
      visit[y] = true;
      edge++;
    }

  return 0;
}
```

OutPut

```
Edge : Weight
0 ---> 5 :  10
5 ---> 4 :  25
4 ---> 3 :  22
3 ---> 2 :  12
2 ---> 1 :  16
1 ---> 6 :  14
```