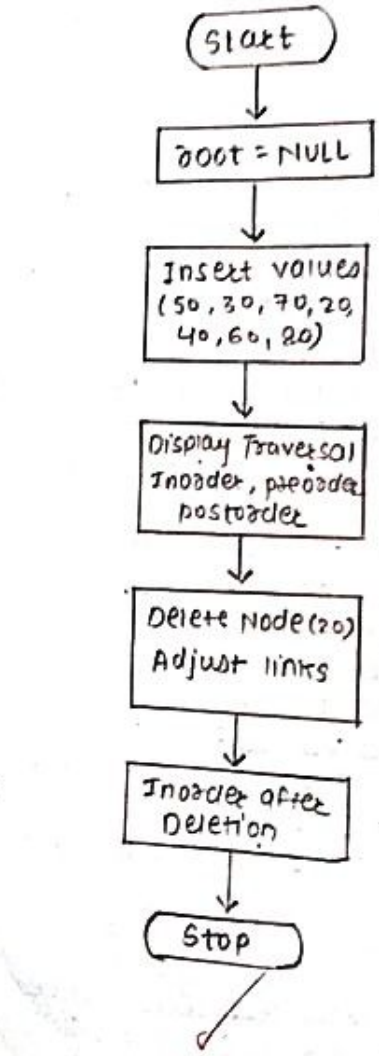
- Insert fixed values : 50, 30, 70, 20, 40, 60, 20
- Display Inorder, preorder and postorder traversal
- Delete node 20.
- Display Inorder traversal again

9. Stop

- Conclusion:

The Binary Search tree was successfully implemented with insertion, deletion, and traversal operations.
It maintains elements in sorted order and allows faster searching with an average time complexity of O(logn)

- Flowchart



Binary search Tree with Insertion, deletion and traversal

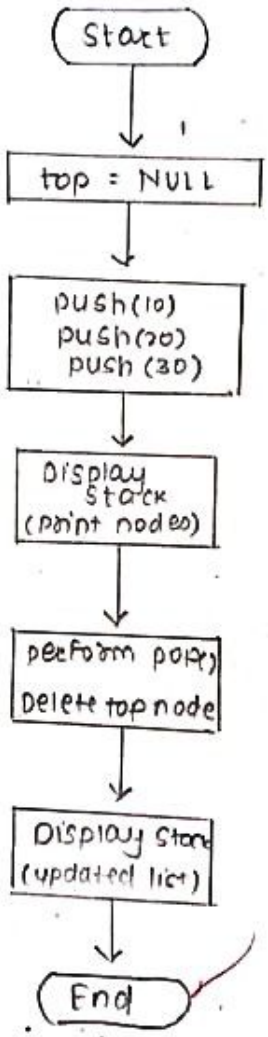
Experiment No. 8

Title: Implement Binary Search Tree with insertion, deletion, and traversal (Inorder, Preorder, Postorder).

Program:

The stack was successfully implemented using both array and linked list. Both implementations perform push and pop operations in constant time O(1). While the array the is simple and faster, the linked list is more flexible because it does not have a fixed size limit

- Algorithm



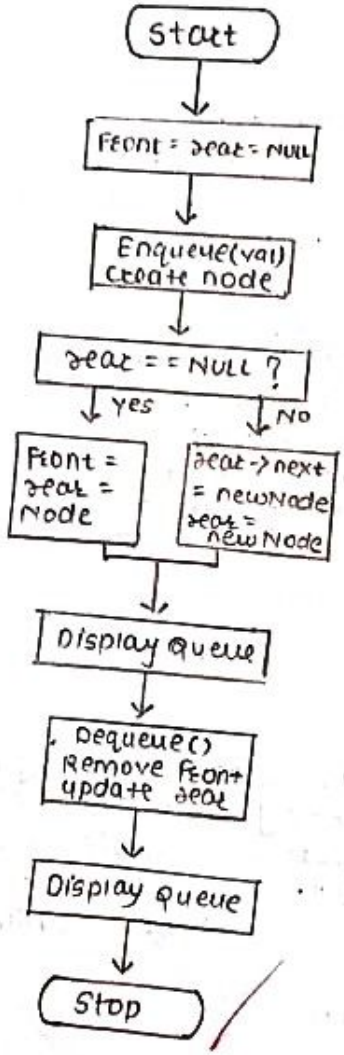
Stack using Linked List

b) Stack using Linked List

Program :

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

struct Node {
  int data;
  Node* next;
};
```

- Flowchart



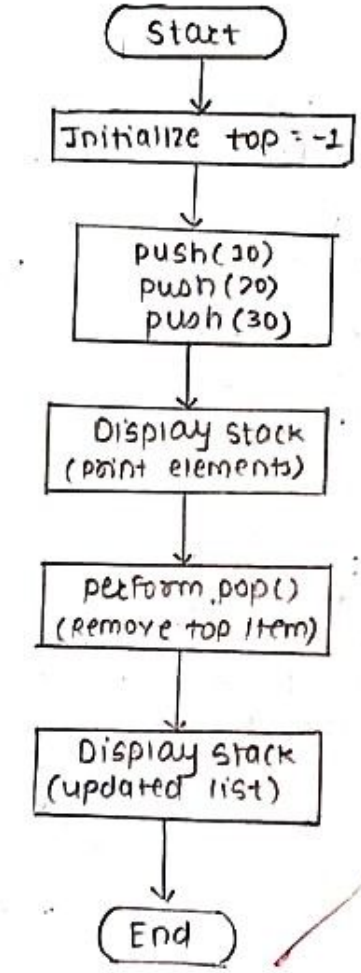
Queue Using Linked List

b) Queue using Linked list

Program :

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

struct Node {
  int data;
  Node* next;
};

class Queue {
  Node* front;
  Node* rear;
public:
  Queue() { front = rear = NULL; }

  void enqueue(int val) {
    Node* newNode = new Node();
    newNode->data = val;
```

• Flowchart



7 / 50

Stack Using Array

Experiment No. 6

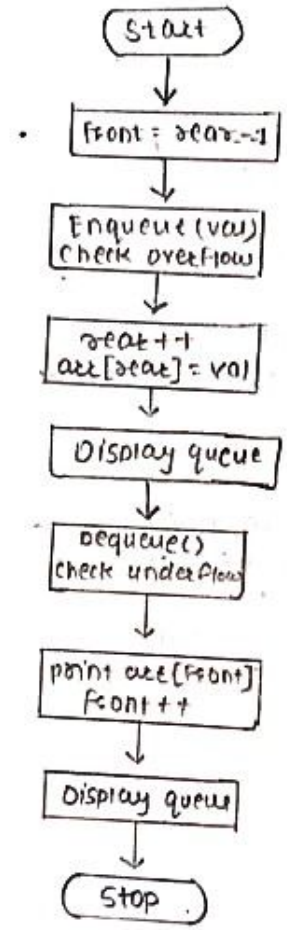
Title: Implement Stack using a) Arrays and b) Linked List.

a) Stack using Array

Program :

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

#define MAX 5

class Stack {
  int arr[MAX];
  int top;

public:
  Stack() { top = -1; }
```

- Flowchart



Queue Using Arrays

Experiment No. 7

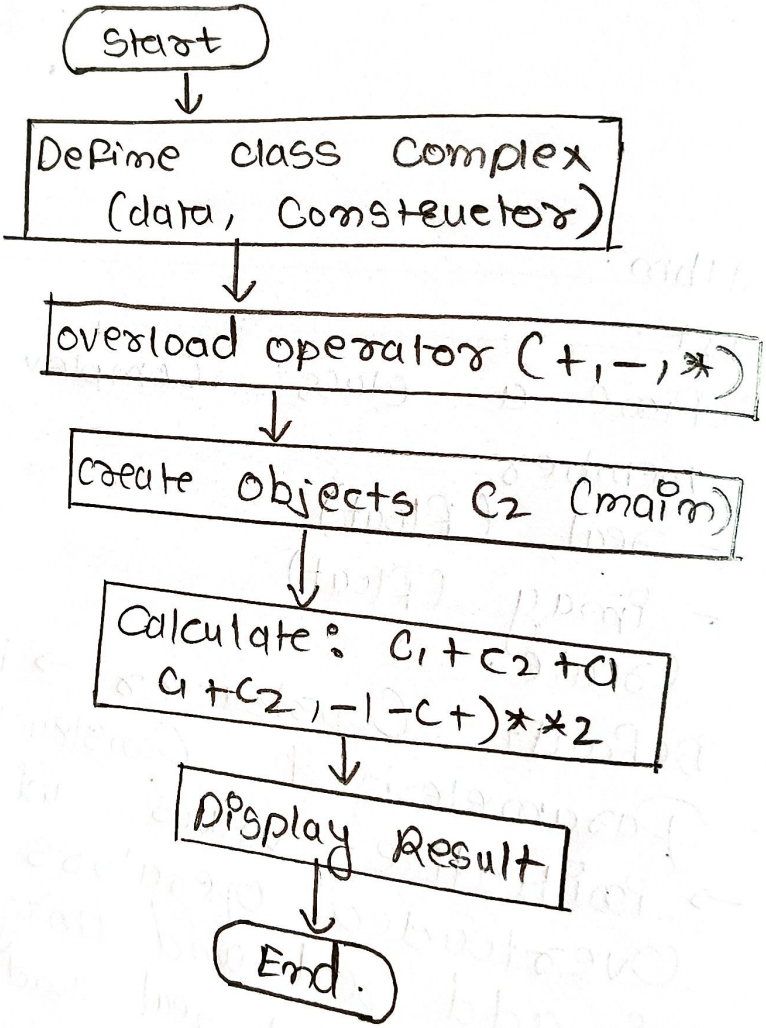
Title: Implement Queue using a) Arrays and b) Linked List.

(a) Queue using Array

Program :

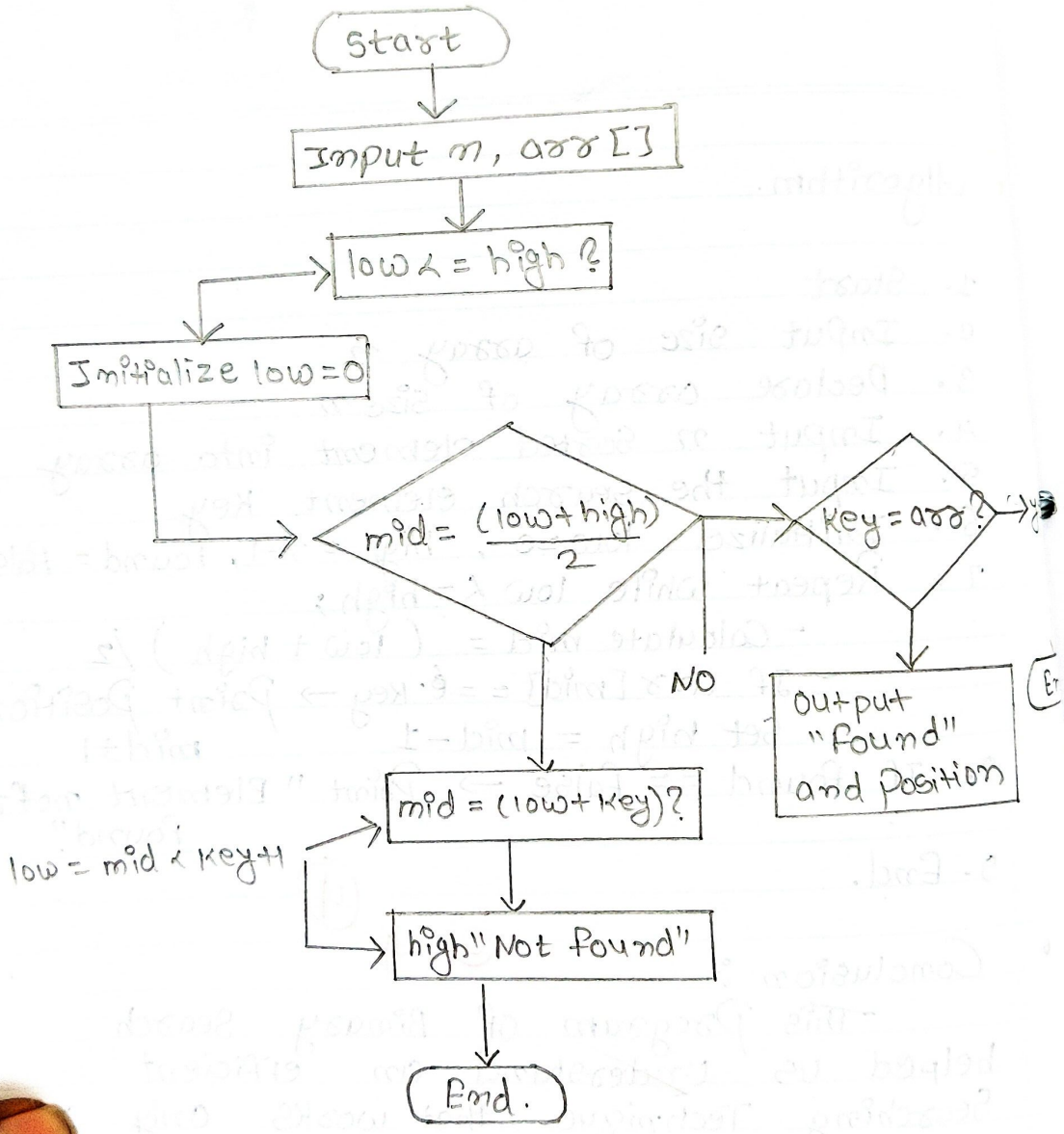
```
#include <iostream>
using namespace std;

#define MAX 5

class Queue {
  int arr[MAX];
  int front, rear;
public:
  Queue() { front = -1; rear = -1; }

  void enqueue(int val) {
    if (rear == MAX - 1)
```

- Flowchart: operator overloading.



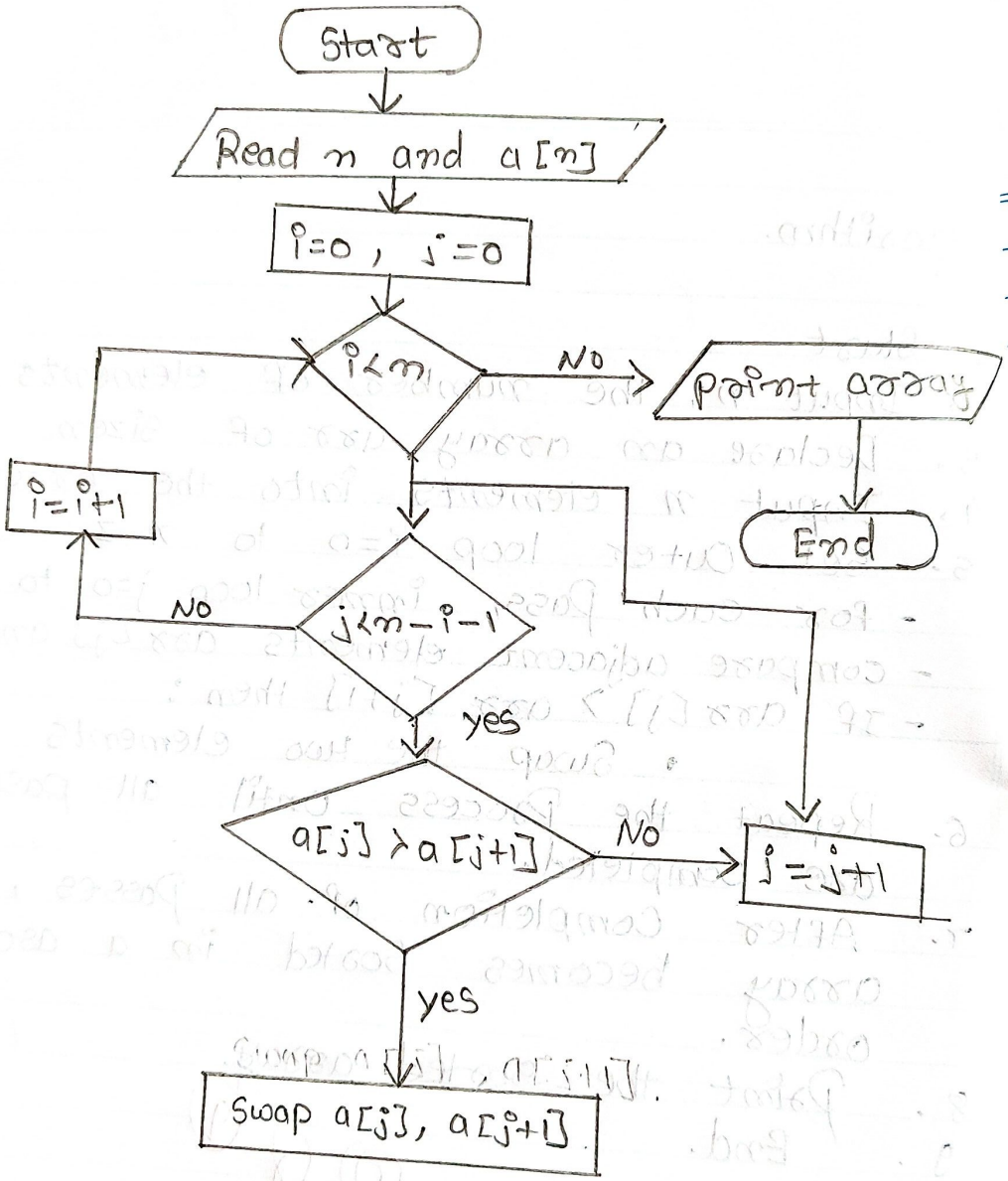
~~Conclusion~~:

- Flowchart.

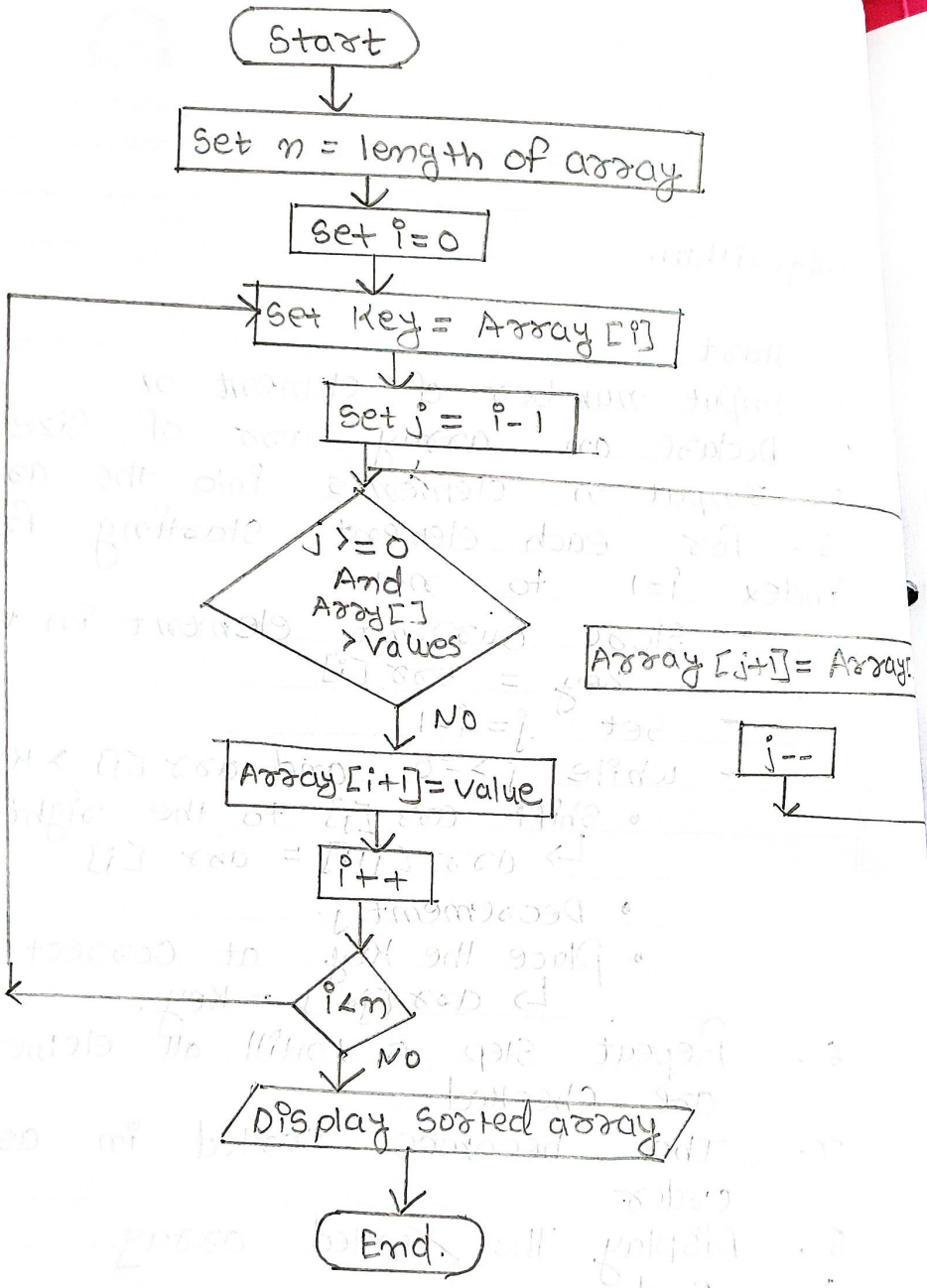


Binary search.

• Flowchart. Bubble Sort.



Flowchart :-



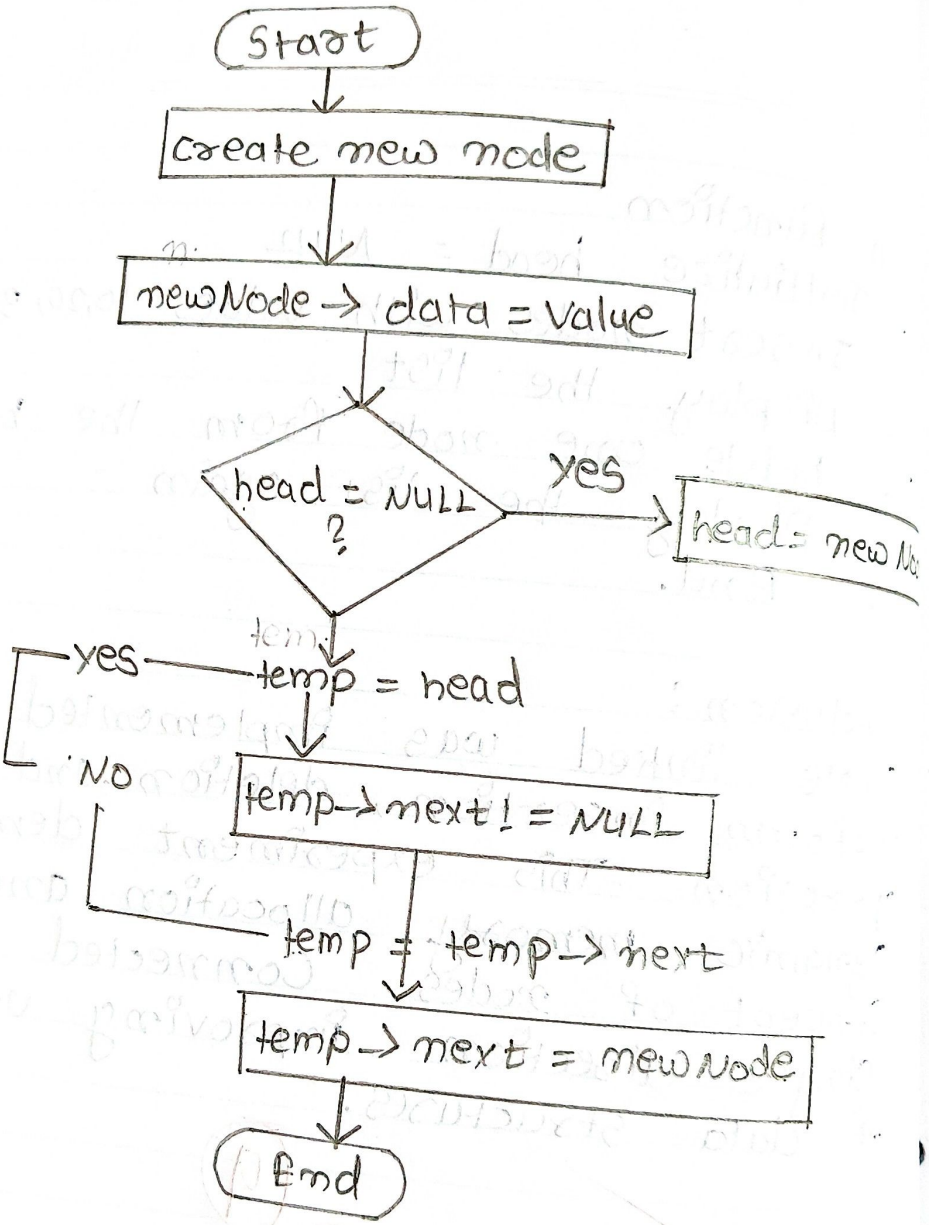
Insertion Sort

Title:
To Implement th

Program:
a) Bubble Sort
#include <iostr
using namespa

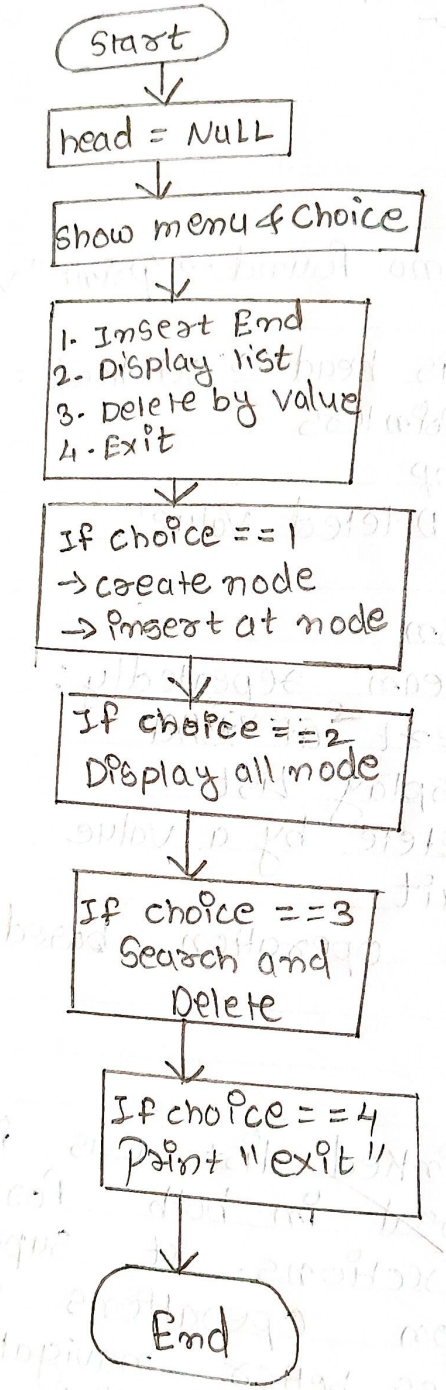
```
void bubbleSo
  for (int i =
    for (int j
      if (arr
        int
        arr
        ar
      }
    }
  }
}

int main()
  int n;
  cout <<
  cin >>
  int arr[

  cout <
  for (i
    cin
  }

  bub

  cou
  for

  }
  co
  re
}

b)
#in
us

v
```

• Flowchart.



Single link List

* Flowchart.



Double linklist

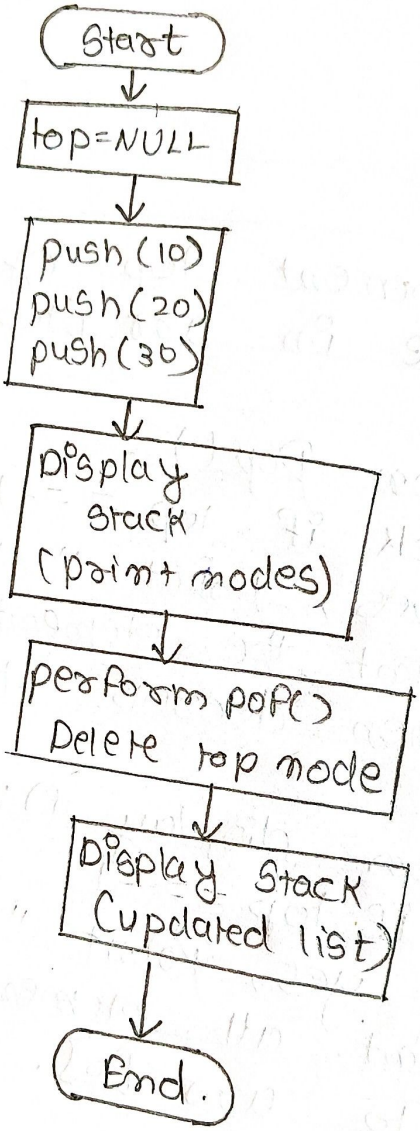
Title:
To implement Doubly linked li

Program:

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

struct Node {
    int data;          // data part
    Node *prev, *next;  // poi
};

Node* head = NULL;  // ini

// Insert node at the end of
void insertEnd(int value) {
    Node* newNode = new
    newNode->data = value
    newNode->next = NUL
    newNode->prev = NUL

    if (head == NULL) {
        head = newNode;
    } else {
        Node* temp = head
        while (temp->next
            temp = temp->n
        temp->next = newl
        newNode->prev =
    }
}

// Display all nodes in
void display() {
    if (head == NULL)
        cout << "List is
        return;
    }
    Node* temp = hea
    cout << "List: ";
```

Flowchart :-

Array / LinkList



Stack Using Linked list:

Experiment

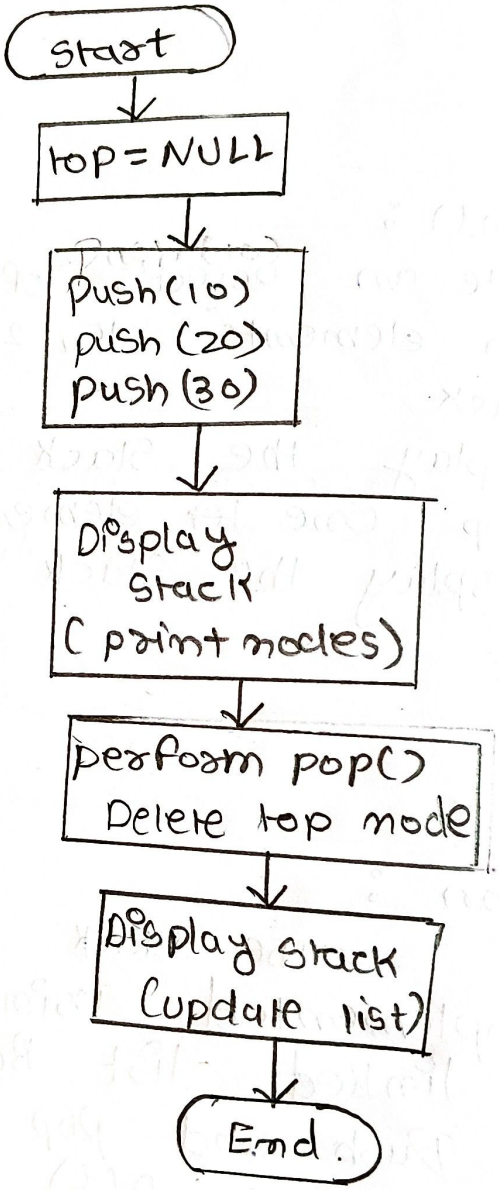
Title: Implen

a)Stack usin

Program :

```
#include <ios
using names

#define MAX

class Stack {
  int arr[MA
  int top;

public:
  Stack() {

  void pus
    if (top
      cou
    else
      arr
  }

  void po
    if (to
      co
    else
      cc
  }

  void
    if (
      c
    els
      c

}
```

- Algorithm :-



Stack Using linked list.

b)Stack using Lir

Program :

```
#include <iostre
using namespac

struct Node {
    int data;
    Node* next;
};

class Stack {
    Node* top
public:
    Stack() { to

    void pus
        Node
        newN
        newN
        top
    }

    void
        if

        e
```