Stack and Queues using Linked List

Dr.B.Saleena

School of Computer Science and Engineering

VIT, Chennai Campus

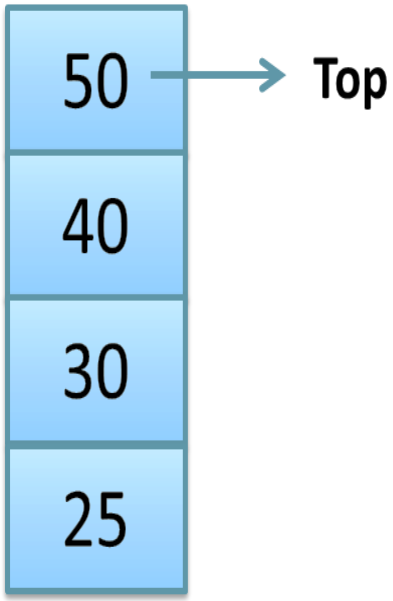
Stack – Definition and Property

- Stack is a linear data structure (ordered list) in which elements are inserted and deleted at one end called the top.
- The last element inserted in to the stack will be the first one to be deleted or removed out of the stack.
- This property is called Last In First Out (LIFO)

Implementation of Stacks

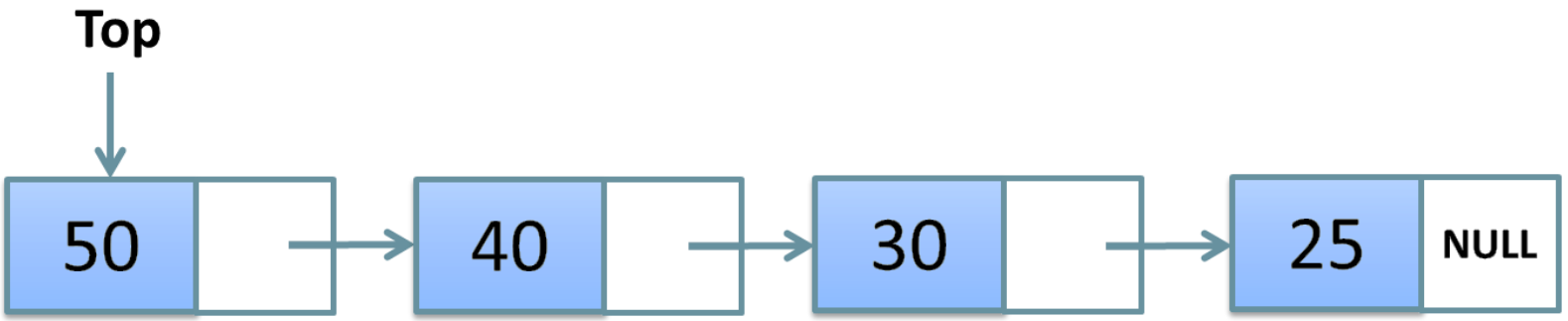
Stacks can be implemented using

- Arrays
- Linked list (Singly or doubly linked list)



Stack – Array Representation

Stack – Singly linked list Representation



Operations of Stack

Push( ) – This operation inserts the element into the top of the stack (Inserting a new node at the front of the linked list)

Pop( ) – This operation deletes the top most element from the stack (Deleting the first node from the linked list)

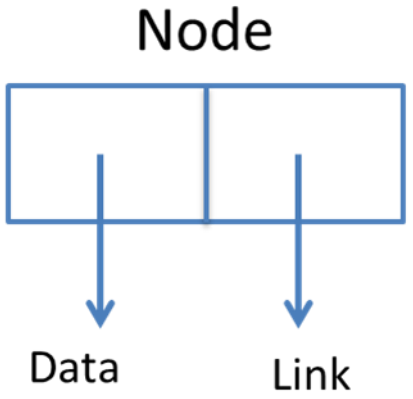
Display( ) – This operation displays the contents of the stack

Definition of a node in stack

A node in a singly linked list with have two fields

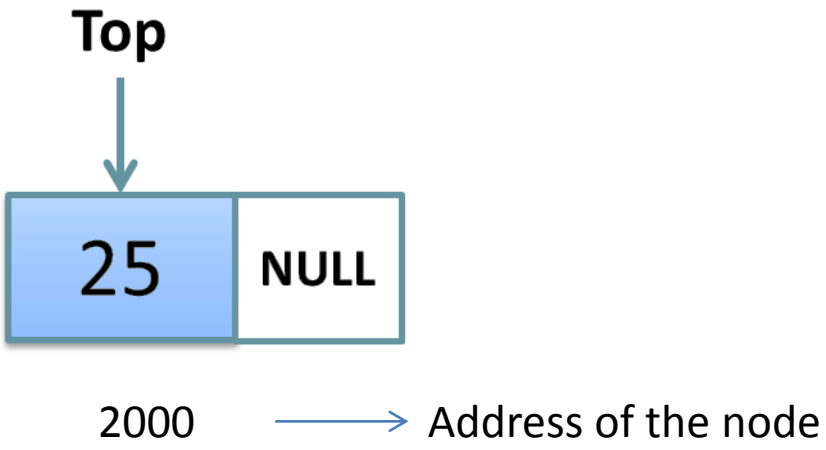
- Data - Holds the value
- Link - Holds the address of the next node

```
struct node
{
int data;  // data field
struct node * link;  // link field
}
```

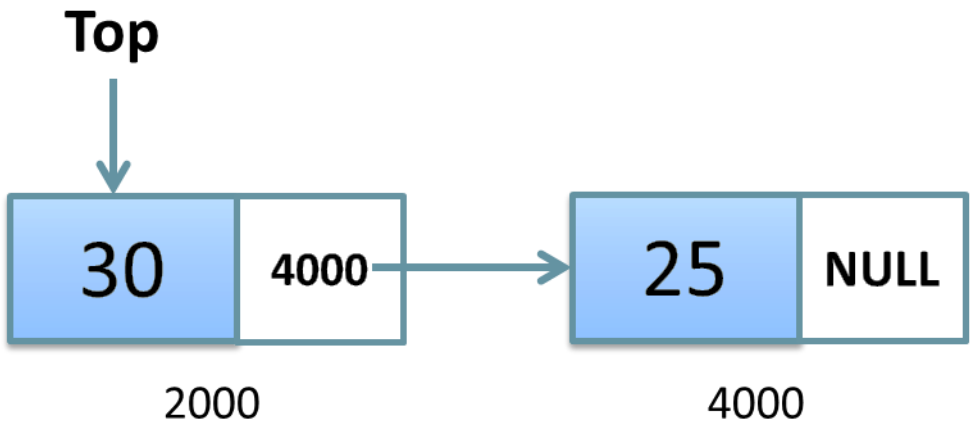


Insertion into the stack (Push)

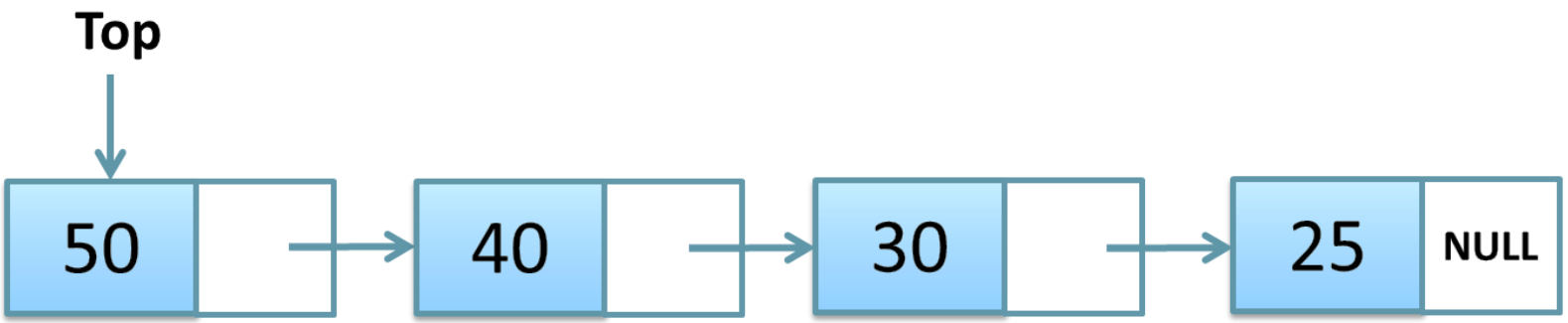
Create a new node and insert data (25) in to the data field and NULL to the link field and name the node as top. (Top is a pointer which points to top most node in the list)



Insert a new data(30) in to the stack. Now top will point to latest element inserted in to the stack



- After Inserting 40 and 50 in to the stack. Now top will point to latest element inserted in to the stack.



Algorithm for Push operation

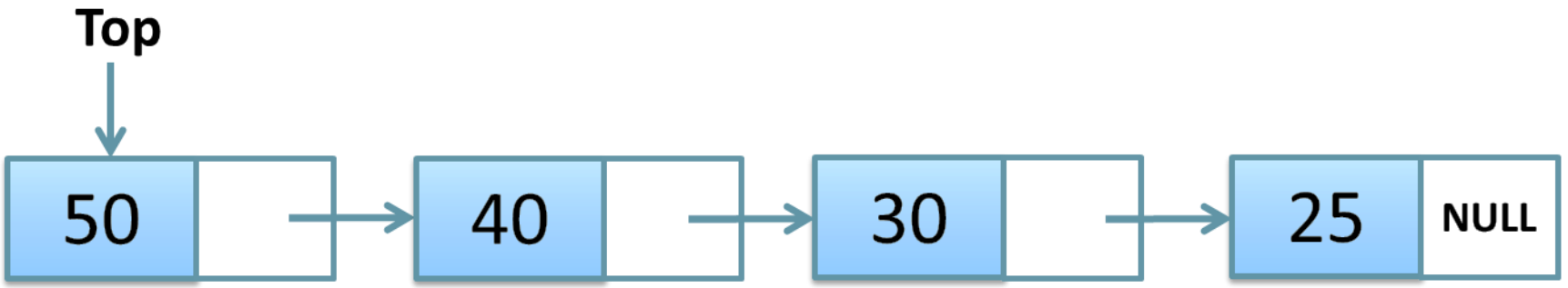
(To insert contents in to the stack)

```
Algorithm push(  item)
{
    newnode =(struct *) malloc (sizeof (struct node));

    newnode->data =item;
    newnode->link= NULL;

    if (top= = NULL)
    top = newnode;
    else
     {
          newnode ->link = top;
          top= newnode;
    }
}
```

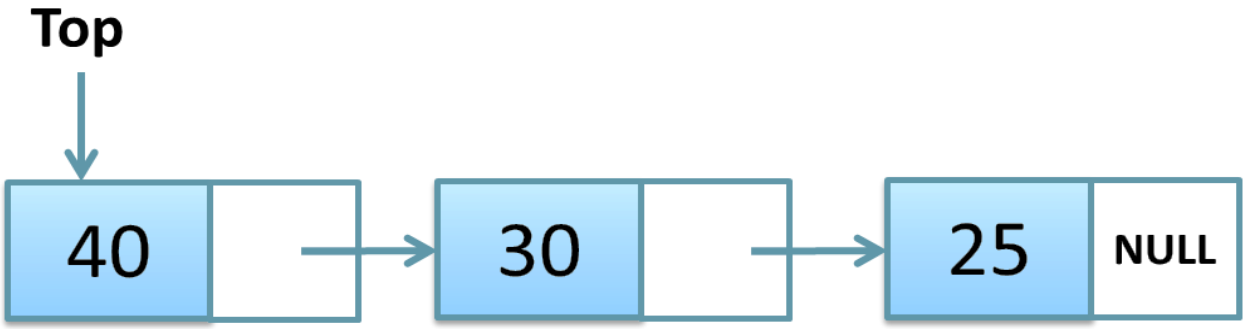
Deletion from the stack (Pop)

Deletes the top most node in the list (where top pointer points to top most item or the latest item inserted into the list)

Original List



After deleting an element from the list



Algorithm for Pop Operation
(To delete the contents from the stack)

```
 Algorithm pop( )
{
   struct node * y;
   if (top = = NULL)
      Print (“STACK IS EMPTY’);
   else
     {
      y = top;
      top= top -> link;
      free(y)
      }
}
```

Algorithm for Display

(To display the contents in the stack)

```
Algorithm display( )
{
struct node * temp;
temp = top;
if (temp = = NULL)
     Print("STACK EMPTY");
else {
          while(temp != NULL)
          {
          print( temp->data);
           temp = temp ->link;
          }
     }
}
```

Queue – Definition and Property

- Queue is a linear data structure (ordered list) in which elements are inserted at one end called the rear and deleted from one end called the front.

- The first element inserted in to the queue will be the first one to be deleted or removed out of the Queue.

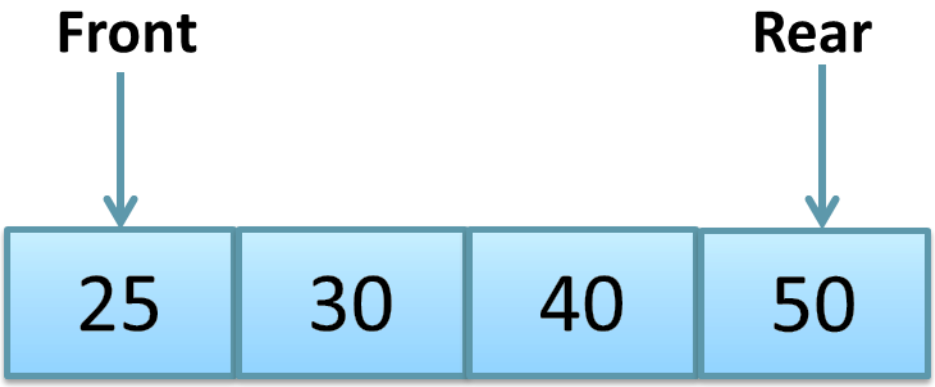
- This property is called First In First Out (FIFO).

Implementation of Queues

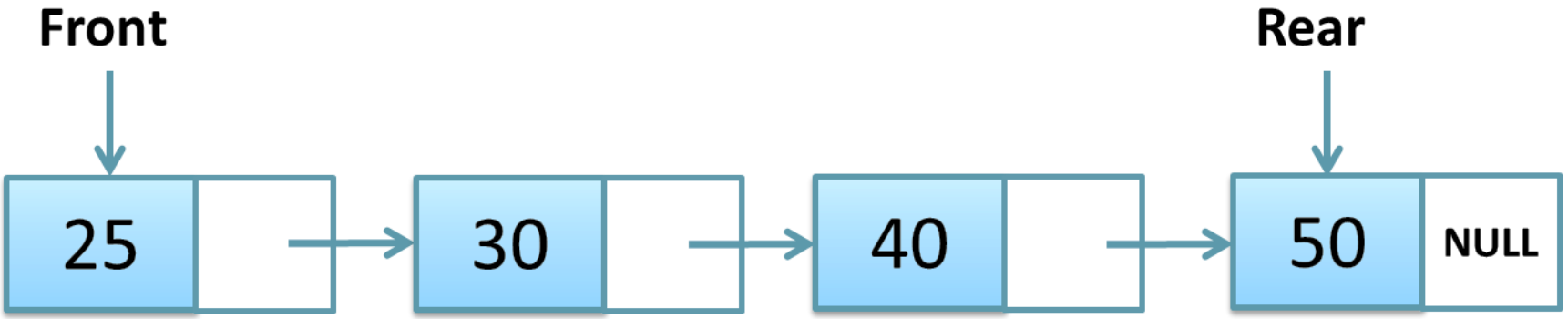
Queues can be implemented using

- Arrays
- Linked list (Singly or doubly linked list)

Queue – Array Representation



Queue – Singly linked list Representation



Operations of Queue

Enqueue( ) – This operation inserts the element into the rear end of the Queue (Insert into the end of the linked list)

Dequeue( ) – This operation deletes the element from the front of the queue. (Deleting the first node from the linked list)

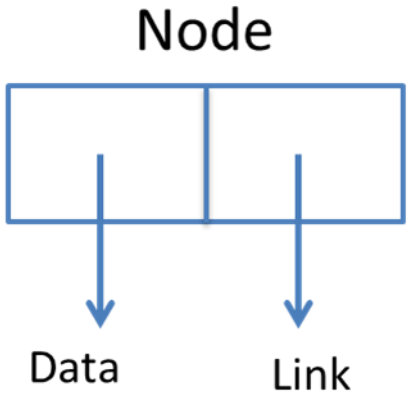
Display( ) – This operation displays the contents of the Queue.

Definition of a node in Queue

A node in a singly linked list with have two fields

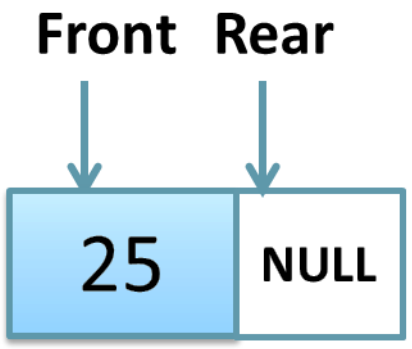
- Data - Holds the value
- Link - Holds the address of the next node

```
struct node
{
int data;  // data field
struct node * link;  // link field
}
```

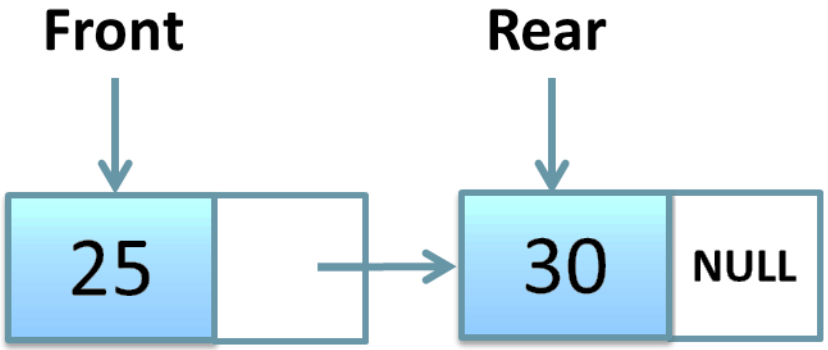


Insertion into the Queue (Enqueue)

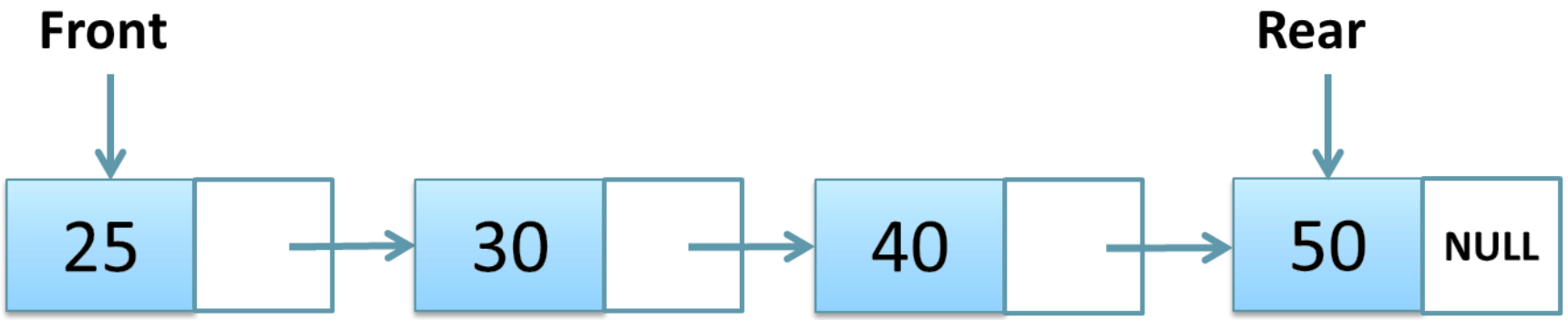
Create a new node and insert data (25) in to the data field and NULL to the link field and name the node as top. (Front points to first element and rear points to last element inserted in to the queue.)



Insert a new data(30) in to the queue. Now front will point to the first element in the list and rear will point to the last element inserted in to the queue.



- After Inserting 40 and 50 in to the queue.



Algorithm for Enqueue Operation

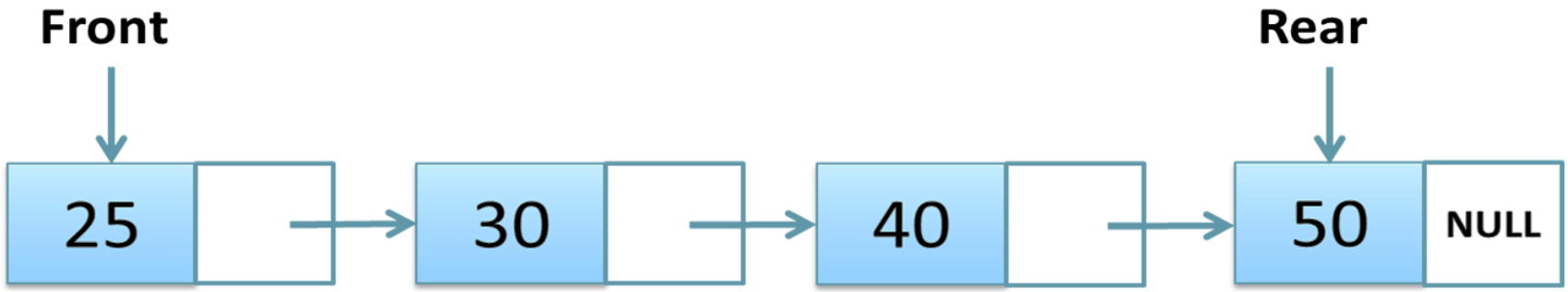
(To insert contents in to the queue)

```
Algorithm enqueue(item)
{
newnode =(struct *) malloc (sizeof (struct node));

newnode->data =item;
newnode->link= NULL;

if (rear= = NULL)&&(front = =NULL)
      rear = newnode ;
      front = newnode;
else
      {
      rear ->link = newnode;
      rear = newnode;
      }
}
```

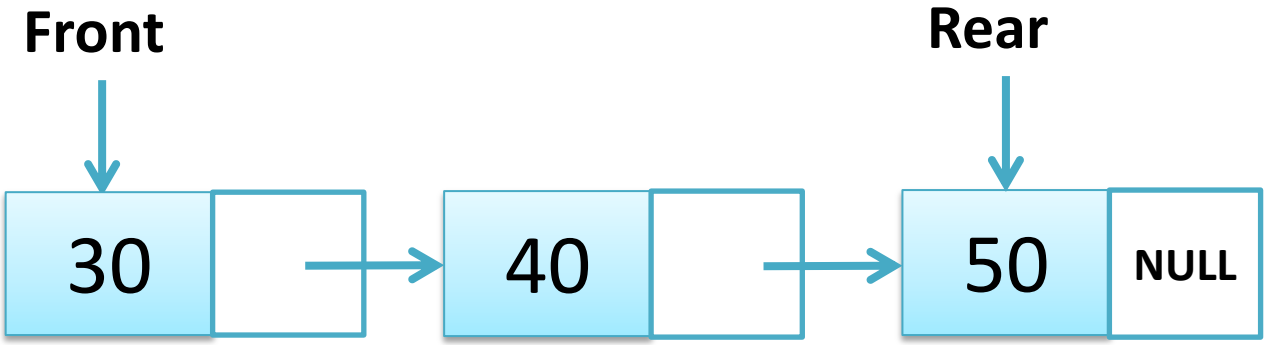
Deletion from the Queue (Dequeue)

Deletes the first node from the list

Original List



After deleting an element from the list



Algorithm for Dequeue Operation

(To delete contents in to the queue)

```
Algorithm dequeue( )
{
struct node * y;

if (front = = NULL)
     Print("QUEUE IS EMPTY');
else
{
     y = front;
     front = front -> link;
     free(y);
}
```

Algorithm to display the Queue

```
void display( )
{
struct node * temp;
temp = front;
if (temp == NULL)
      Print("QUEUE EMPTY");
else
{
while(temp != NULL)
      {
      Print ( temp->data);
      temp = temp ->link;
      }
}
}
```

Thank you