Node Class


```
class Node
{
    private:
        int Data;
        Node* Next;
        public:
       Node();
       int GetData();
       Node* GetNext();
       void SetData(int);
       void SetNext(Node*);
};
```

Constructor


```
Node::Node()
{

}
```

Getters


```
int Node::GetData()
{
    return Data;
}

Node* Node::GetNext()
{
     return Next;
}
```

Setters


```
void Node::SetData(int Data)
{
    this->Data = Data;
}

void Node::SetNext(Node* Next)
{
    this->Next = Next;
}
```

List Class

List Class


```
class List {
        private:
                Node *CurrentLocation;
        public:
                List();
                void Print();
                void Insert(int data);
                void Delete(int data);
};
```

Constructor


```
List()
{
        CurrentLocation = NULL;
}
```

Print Member Function


```
void List::Print() {

    // Temp Node Pointer
    Node *tempNode = CurrentLocation;

    // Case1: Empty List
    if ( tempNode == NULL ) {
        cout << "List is Empty" << endl;
        return;
    }
```

Print Member Function


```
// Case2: Only one Node in the List

    if ( tempNode->GetNext() == NULL ) {
        cout << tempNode->GetData();
        cout << " --> ";
        cout << "NULL" << endl;
    }
```

Print Member Function


```
Case3: More than one Node in the List
else {
        do {
            cout << tempNode->GetData();
            cout << " --> ";
            tempNode = tempNode->GetNext();
    }
    while ( tempNode != NULL );

    cout << "NULL" << endl;
    }
}//End of Print()
```

Insert (end) Member Function


```
void List::Insert(int data) {

    // Create a new Node
    Node* newNode = new Node();
    newNode->SetData(data);
    newNode->SetNext(NULL);
```

Insert (end) Member Function


```
// Create a temp pointer
    Node *tempNode = CurrentLocation;

    if ( tempNode != NULL ) {
    // Nodes already present in the list
    // Parse to end of list
    while ( tempNode->GetNext() != NULL ) {
        tempNode = tempNode->GetNext();
    }
```

Insert (end) Member Function


```
// Point the last node to the new Node
    tempNode->SetNext(newNode);
    }
    else {
    // First node in the list
    CurrentLocation = newNode;
    }
}//End of Insert
```

Delete Member Function


```
void List::Delete(int data) {

    // Create a temp pointer
    Node *tempNode = CurrentLocation;

    // Case1: No nodes
    if ( tempNode == NULL )
    return;

    // Case2: Last node of the list
    if ( tempNode->GetNext() == NULL ) {
    delete tempNode;
    CurrentLocation = NULL;
    }
```

Delete Member Function


```
Case3: More than one Node
else {
    // Parse through the nodes
    Node *previous;
    do {
        if ( tempNode->GetData() == data ) break;
         previous = tempNode;
         tempNode = tempNode->GetNext();
    } while ( tempNode != NULL );

    // Adjust the pointers
    previous ->SetNext(tempNode->GetNext());

    // Delete the current node
    delete tempNode;
    tempNode = NULL;
    }
```