Logical Explanation of the Code

Introduction:

This C++ program implements a **singly linked list** with functions to insert nodes at the **beginning** and **end**, and display the list.

1. Inserting at the Beginning (push_front())

- Creates a new node with the given value.
- If the list is empty, the new node becomes both **head** and **tail**.
- Otherwise, the new node points to the current **head**, and **head** is updated.

2. Inserting at the End (push_back())

- Creates a new node with the given value.
- If the list is empty, the new node becomes both **head** and **tail**.
- Otherwise, the current **tail**'s next pointer is set to the new node, and **tail** is updated.

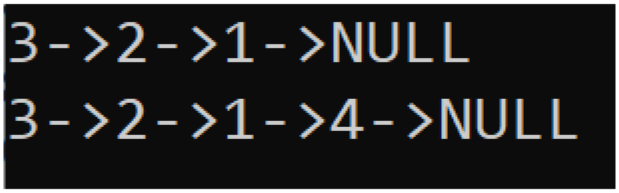
3. Displaying the List (DisplayLL())

- Starts from **head** and prints each node's value.
- Stops when reaching NULL, indicating the end of the list.

4. Function Calls & Execution:

- Nodes **3, 2, and 1** are inserted at the beginning.
- The list is displayed: **3->2->1->NULL**.
- Node **4** is inserted at the end.
- The updated list is displayed: **3->2->1->4->NULL**.

Final Output:



```
3->2->1->NULL
3->2->1->4->NULL
```

Code:

```
#include <iostream>

using namespace std;

class Node

{

    public:

        int data;

        Node* next;

    Node(int val)

    {

        data = val;

        next = NULL;

    }

};


class List

{

    Node* head;

    Node* tail;


    public:

        List()

        {

            head = tail = NULL;

        }


        void push_front(int val)
```

```
{
    Node* newNode = new Node(val);

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

void push_back(int val)
{
    Node* newNode = new Node(val);

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

```
        }

        void DisplayLL()
        {
            Node* temp = head;
            while(temp != NULL)
            {
                cout << temp->data << "->";
                temp = temp->next;
            }
            cout << "NULL" << endl;
        }
};

int main()
{
    List ll;
    ll.push_front(1);
    ll.push_front(2);
    ll.push_front(3);
    ll.DisplayLL();
    ll.push_back(4);

    ll.DisplayLL();
    return 0;
}
```