

|  |
| --- |
| **LAB 3** of DSA LAB |

**Submitted by:**

**Name:** Muhammad shuraim

**Roll no:** SU92-S24-BSSEM-103

**Section:** 3A

**Submitted to**

Sir Rasikh Ali

**Question 1**

#include <iostream>

using namespace std;

class Node {

public:

int value;

Node\* next;

Node(int val) {

value = val;

next = nullptr;

}

};

class LinkedList {

public:

Node\* head;

LinkedList() {

head = nullptr;

}

void insert\_at\_start(int val) {

Node\* newnode = new Node(val);

newnode->next = head;

head = newnode;

}

void insert\_at\_last(int val) {

Node\* newnode = new Node(val);

if (head == nullptr) {

head = newnode;

return;

}

Node\* temp = head;

while (temp->next != nullptr) {

temp = temp->next;

}

temp->next = newnode;

}

void display() {

if (head == nullptr) {

cout << "List is empty!" << endl;

return;

}

Node\* temp = head;

while (temp != nullptr) {

cout << temp->value << " -> ";

temp = temp->next;

}

cout << "NULL" << endl;

}

};

int main() {

LinkedList obj;

obj.insert\_at\_start(1);

obj.display();

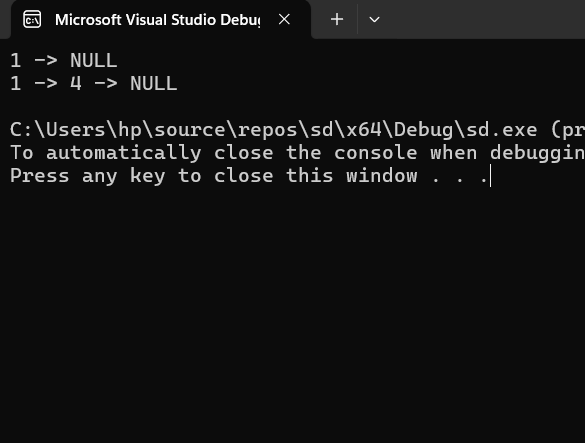
obj.insert\_at\_last(4);

obj.display();

return 0;

}

**OUTPUT**



Explanation

* First I create class node which contain int value and pointer next which point to next node and constructor
* Then class linkedlist which contain head (first value) and constructor
* Insert at start function

1. Create a new node value=val
2. Set newnode->next to head
3. Move head to point new node

Insert at end

* If list is empty head=newnode
* Otherwise traverse to the last node
* Lastnode->=newnode

Dislay