

DATA Lab

Lab-4



Name: Talib Husain

Roll# 21F-9070

**Task-4**

#include<iostream>

using namespace std;

struct node {

int data;

node\* next;

};

class list {

node\* head;

static int count;

public:

list(node \*h = NULL) {

head = h;

}

bool insert(int value,int pois) {

if (pois <= 0) return false;

int currIndex = 2;

node\* current = head;

while (current && pois > currIndex) {

current = current->next;

currIndex++;

}

if (pois > 1 && current == NULL) return false;

node\* newNode = new node;

newNode->data = value;

if (pois == 1) {

newNode->next = head;

head = newNode;

}

else {

newNode->next = current->next;

current->next = newNode;

}

return true;

}

void con\_to\_circular() {

node\* temp = head;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = head;

}

void display() {

node\* current = head;

while (current->next != NULL && current->next!=head) {

cout << current->data << "->";

current = current->next;

}

cout << current->data << "\n";

}

};

int main() {

list l;

cout << l.insert(1,1) << endl;

cout << l.insert(2, 2) << endl;

cout << l.insert(3, 3) << endl;

cout << l.insert(4, 4) << endl;

cout << l.insert(5, 5) << endl;

l.display();

l.con\_to\_circular();

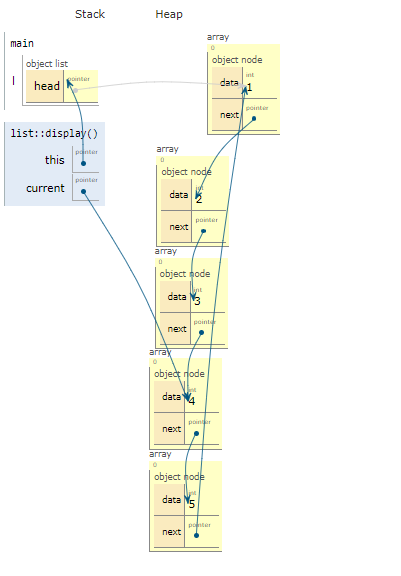
cout << "After converting in circular\n";

l.display();

}

**Text

Description automatically generated**



**Task-5**

#include <iostream>

using namespace std;

// Class for Node

class Node

{

public:

int data;

Node\* next;

};

//class list for implementation of Linked list

class List {

Node\* head;

Node\* next;

public:

List(Node \*h = NULL) {

head = h;

next = NULL;

}

// For insert a node at the beginning

void InsertAtBegin(int data)

{

// New node

Node\* newNode = new Node;

newNode->data = data;

newNode->next = head;

// Set the null if the link is not NULL

if (head != NULL)

{

Node\* temp = head;

while (temp->next != head)

temp = temp->next;

temp->next = newNode;

}

else

// For first node

newNode->next = newNode;

head = newNode;

}

//code for deleting at end

void DeleteAtEnd() {

Node\* current = head;

Node\* temp = NULL;

//Moving to end of list

while (current->next != head && current) {

if (current->next->next)

temp = current;

current = current->next;

}

if(temp)

temp->next = head;

delete current;

current = NULL;

}

// For printing

void printList()

{

Node\* temp = head;

while (temp->next!=head)

{

cout<<temp->data<<' ';

temp = temp->next;

}

cout << temp->data << ' ';

cout << endl;

}

};

// Main

int main()

{

List l;

l.InsertAtBegin(1);

l.InsertAtBegin(2);

l.InsertAtBegin(3);

l.InsertAtBegin(4);

cout << "Before Deletion: ";

l.printList();

l.DeleteAtEnd();

cout << "After Deletion: ";

l.printList();

}

**Text

Description automatically generated**