

DATA Lab

Lab-4



Name: Talib Husain

Roll# 21F-9070

**Task-4**

#include<iostream>

using namespace std; struct node {

int

data; node\* next;

}; class linked\_list {

node\* head; static int count; public:

linked\_list() {

head = NULL;

} bool is\_empty() {

if (head == NULL) {

return true;

}

else {

return false;

}

} void create\_node(int value) {

if (is\_empty() == true) {

node\* newnode; node\* cur;

newnode = new node();

cout << "The first value is" << endl; cin >> newnode->data;

count++; head = newnode; cur = newnode;

for (int i = 1; i < value; i++)

{

cur->next = new node(); cur = cur->next;

cout << "Enter the value at " << i + 1 << " index" << endl; cin >> cur->data;

count++;

}

cur->next = NULL; cout << endl;

}

}

void cov\_into\_circular() {

node\* temp = head; while (temp -

> next != NULL) {

temp = temp->next;

}

temp->next = head;

} void print() {

node\* temp = head; while (temp->next != head) {

cout <<

temp->data << " "; temp = temp->next;

}

cout << temp->data << endl;

}

};

int linked\_list::count = 0; int main() {

linked\_list obj; int num;

cout << "enter the number of nodes:" << endl; cin >> num;

obj.create\_node(num);

obj.cov\_into\_circular(); obj.print();

return 0;

}

**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**