

**DSA**

**Lab Journal: 04**

**Task : 02**

**Exercise: 02**

**Link:**

https://github.com/FaisalTayyab/LJ4Task2Exercise2.git

**Code:**

#include<iostream>

using namespace std;

//creating a class node

class node {

public:

int data;

node\* next;

};

//creating a class for circulaer linked list

class Clist

{

private:

node\* head;

public:

Clist()

{

head = NULL;

}

bool is\_empty()

{

if (head == NULL)

{

return true;

}

else {

return false;

}

}

//creating function for inserting node at begin

void insert\_begin(int value)

{

node\* temp = new node;

temp->data = value;

temp->next = NULL;

if (head == NULL)

{

head = temp;

head->next = head;

}

node\* ptr = head;

while (ptr->next != head)

{

ptr = ptr->next;

}

ptr->next = temp;

temp->next = head;

head = temp;

}

//creating function for inserting node at end

void insert\_end(int value)

{

node\* ptr = new node;

ptr->data = value;

ptr->next = NULL;

if (head == NULL)

{

head = ptr;

ptr->next = head;

}

else

{

node\* temp = head;

while (temp->next != head)

{

temp = temp->next;

}

temp->next = ptr;

ptr->next = head;

}

}

//function for node insert after value

void insert\_after(int OldV, int NewV)

{

node\* temp = head;

if (head == NULL)

{

cout << "List is empty"; return;

}

else if (head != NULL) {

if (head->data == OldV)

{

node\* t = new node; t->data = NewV; t->next = temp->next;

head->next = t;

}

else if (head->data != OldV)

{

do

{

temp = temp->next;

}

while(temp->data != OldV && temp != head);

if (temp->data == OldV)

{

node\* t = new node;

t->data = NewV; t->next = temp->next;

temp->next = t;

}

else {

cout << "Node not found";

return;

}

}

}

}

//function for deleting a node

void delete\_begin()

{

node\* temp;

temp = head;

if (head == NULL) {

cout << "List has no nodes";

return;

}

//Only one node in list

if (head->next == head) {

head = NULL;

delete temp;

return;

}

else {

node\* t = head;

while (temp->next != head)

{

temp = temp->next;

}

head = head->next;

temp->next = head;

delete t;

}

}

//function for deleting node in list

void delete\_end()

{

node\* temp;

temp = head;

if (head == NULL) {

cout << "\nList has no nodes";

return;

}

//Only one node in list

if (head->next == head) {

head = NULL;

delete temp;

return;

}

else {

node\* t = NULL;

while (temp->next != head)

{

t = temp;

temp = temp->next;

}

t->next = temp->next;

delete temp;

}

}

//function for deleting specific nopde

void deletenode(int value)

{

node\* temp = head, \* t = NULL;

if (head == NULL) {

cout << "List is empty"; return;

}

else {

if (head->data == value)

{

do {

t = temp;

temp = temp->next;

} while (temp != head);

head = head->next;

delete temp;

t->next = head;

}

else {

do {

t = temp;

temp = temp->next;

} while (temp->data != value && temp != head);

if (temp != head) {

t->next = temp->next;

delete temp;

}

else {

cout << "Node not found"; return;

}

}

}

}

//display function that displays the values

void display()

{

node\* current = head;

if (head == NULL)

{

cout << "List has no element!!" << endl;

}

do

{

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

current = current->next;

} while (current != head);

cout << endl;

}

};

//main file

int main()

{

Clist li;

li.insert\_begin(10);

li.insert\_begin(20);

li.insert\_begin(30);

li.insert\_begin(40);

li.insert\_begin(50);

li.insert\_end(60);

li.insert\_end(70);

li.insert\_end(80);

li.delete\_begin();

li.delete\_end();

li.deletenode(30);

li.display();

}

**Output:**

