# Mahad Ghauri

233523

DSA Lab Task

#ifndef NODE\_H

#define NODE\_H

#include <iostream>

#include <string>

using namespace std;

class Node

{

private:

    int data;

    Node \*next;

public:

    Node(int x)

    {

        data = x;

        next = NULL;

    }

    void setNext(Node \*other)

    {

        next = other;

    }

    void setData(int x)

    {

        data = x;

    }

    Node \*getNext()

    {

        return next;

    }

    int getData()

    {

        return data;

    }

};

#endif

#ifndef LIST\_H

#define LIST\_H

#include "NODE.H"

class List

{

public:

    Node \*head = NULL;

    Node \*tail = NULL;

    List()

    {

        head = NULL;

        tail = NULL;

    }

    void AddNode(int value)

    {

        if (head == NULL)

        {

            head = new Node(value);

            tail = head;

        }

        else

        {

            tail->setNext(new Node(value));

            tail = tail->getNext();

        }

    }

    void Add\_NewNode(int value)

    {

        Node \*addingNode = new Node(value);

        if (head == NULL)

        {

            head = addingNode;

            tail = addingNode;

            tail->setNext(head);

        }

        else

        {

            tail->setNext(addingNode);

            tail = tail->getNext();

            tail->setNext(head);

        }

    }

    void Display()

    {

        Node \*temp = head;

        cout << " List: ";

        while (temp != NULL)

        {

            cout << temp->getData() << " ";

            temp = temp->getNext();

        }

        cout << endl;

    }

    List \*CircularList()

    {

        if (head == NULL)

        {

            return this;

        }

        tail->setNext(head);

        return this;

    }

    void Circular\_Display()

    {

        Node \*temp = head;

        cout << " List: ";

        do

        {

            cout << temp->getData() << " ";

            temp = temp->getNext();

        } while (temp != head);

        cout << endl;

    }

    bool Check\_Node(int value)

    {

        bool found = false;

        Node \*current = head;

        while (current != NULL)

        {

            if (current->getData() != value)

            {

                current = current->getNext();

            }

            else

            {

                found = true;

                break;

            }

        }

        return found;

    }

    bool Check\_Circular\_List()

    {

        bool found = false;

        Node \*current = head;

        while (current->getNext() != NULL)

        {

            if (current->getNext() == NULL)

            {

                found = true;

                return found;

            }

            else

            {

                return found;

            }

            current = current->getNext();

        }

        return found;

    }

    int getSize()

    {

        int count = 0;

        Node \*temp = head;

        do

        {

            count++;

            temp = temp->getNext();

        } while (temp != head);

        return count;

    }

    List \*Move\_End()

    {

        Node \*current = head;

        Node \*previous = NULL;

        Node \*last = tail;

        int count = getSize();

        int array[count];

        current = head;

        for (int i = 0; i < count; i++)

        {

            array[i] = current->getData();

            current = current->getNext();

        }

        int arr[count];

        Node \*temp = head;

        for (int i = 0; i < count; i++)

        {

            if (current->getData() == current->getData())

            {

                temp = current;

                previous->setNext(current->getNext());

                delete current;

                last->setNext(temp);

                temp->setNext(head);

            }

            else

            {

                current = current->getNext();

            }

        }

        return this;

        return this;

    }

};

#endif

#include "NODE.H"

#include "LIST.H"

int main()

{

    List firstList;

    int oddArray[] = {1, 3, 5, 7 ,9  ,7 , 11 };

    for (int i = 0; i < 7; i++)

    {

        firstList.AddNode(oddArray[i]);

    }

    firstList.Display();

    List \*circularList = firstList.CircularList();

    cout << "\n Adding a new element in the circular linked list: " << endl;

    circularList->Add\_NewNode(13);

    circularList->Circular\_Display();

    cout << "\n";

    int position = 3, value = 5;

    cout << "\n Size of the list is " << circularList->getSize() << endl;

    cout << "\n Updating Node : " << endl;

    bool found = circularList->Check\_Node(value);

    if (found)

    {

        cout << " The value " << value << " is present  in the list " << endl;

    }

    cout << "\n";

    cout << "\n Check circular list : " << endl;

    bool choice = circularList->Check\_Circular\_List();

    if (choice)

    {

        cout << " The list is not circular list "  << endl;

    }

    else

    {

        cout << " The list is circular " << endl;

    }

    cout << "\n";

    cout << "\n Moving the elements to the end of the list " << endl;

    List\* movingList = circularList->Move\_End();

    movingList->Circular\_Display();

}

