Name: **Muhammad Anas Baig**

Enrollment No.: **01-134152-037**

Section: **BS(CS)-4A**

LAB-JOURNAL-8

Exercise 1:

Implement the (class) Circular Linked List to create a list of integers. You need to provide the implementation of the member functions as described in the following.

```
class CList
{
private:
      Node * head;
public:
      CList();

      // Checks if the list is empty or not
      bool emptyList();

      // Inserts a new node with value 'value' at position 'pos'
      in the list
      void insert (int pos, int value);

      // Inserts a new node at the start of the list
      void insert_begin(int value);

      // Inserts a new node at the end of the list
      void insert_end(int value);
```

```
      // Deletes a node from position 'pos' of the list
      void deleteNode(int pos);

      // Deletes a node from the beginning of the list
      void delete_begin();

      // Deletes a node from the end of the list
      void delete_end();

      // Displays the values stored in the list
      void traverse();
};
```

Solution:

Node.h File:

```cpp
#pragma once
class Node
{
public:
    int data;
    Node *next;
public:
    Node(void);
};
```

Node.cpp File:

```cpp
#include "Node.h"

Node::Node(void)
{
}
```

cList.h File:

```cpp
#include "Node.h"
#pragma once
class cList
{
public:
    Node *head;
public:
    cList(void);
    bool isEmpty();

    void beginInsert(int);
    void midInsert(int, int); //count nodes and insert Node next to count == Z
    void endInsert(int);

    void beginDelete();
    void midDelete(int); //delete node from position X
    void endDelete();

    void displaycList();
};
```

cList.cpp File:

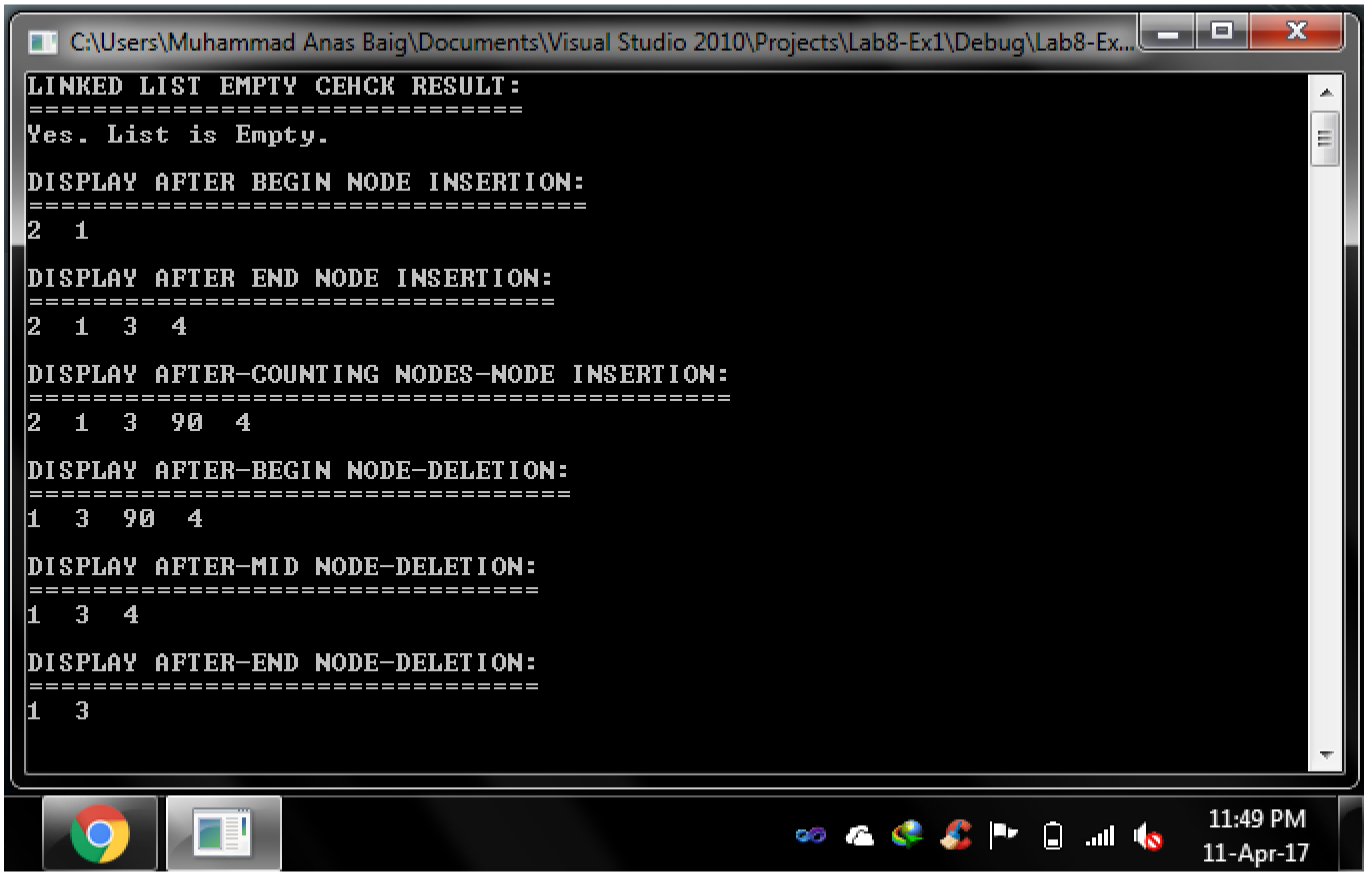
```cpp
#include "cList.h"
#include "conio.h"
#include "Node.h"
#include <iostream>
using namespace std;

cList::cList(void)
{
    head = '\0';
}

bool cList::isEmpty()
```

```cpp
{
    if(head == '\0')
    {
        return true;
    }
    else
    {
        return false;
    }
}

void cList::beginInsert(int value) //inserts node at beginning
{
    Node *ptr;
    ptr = new Node;
    ptr->data = 0;
    ptr->next = '\0';

    ptr->data = value;
    ptr->next = head;

    if( !isEmpty() )
    {
        Node *temp = head;
        while( temp->next != head)
        {
            temp = temp->next;
        }
        temp->next = ptr;
    }
    else
    {
        ptr->next = ptr;
    }
    head = ptr;
}

void cList::midInsert(int value, int countLocation) //count Nodes and insert Node next to count==Z
{
    if(!isEmpty())
    {
        Node *temp = head;

        for( int i=1; i<countLocation; i++ )
        {
            temp = temp->next;
            if( temp == '\0' )
            {
                cout<<"Sorry!!! No Node found at Count = "<<countLocation<<"."<<endl;
                return;
            }
        }
        Node *ptr = new Node;
        ptr->data = value;
        ptr->next = temp->next;
        temp->next = ptr;
    }
    else
    {
        cout<<"Sorry!!! No data inserted-No count found-cList is empty"<<endl;
    }
}

void cList::endInsert(int value)
{
    Node *ptr;
    ptr = new Node;
    ptr->data = 0;
    ptr->next = '\0';

    ptr->data = value;

    if( !isEmpty() )
    {
        Node *temp = head;
        while( temp->next != head)
        {
            temp = temp->next;
        }
        temp->next = ptr;
        ptr->next = head;
    }
    else
    {
        ptr->next = ptr;
        head = ptr;
    }
    }

    void cList::beginDelete()
    {
        if(!isEmpty())
        {
            Node *temp = head;
            Node *temp1 = head;

            if( head->next == head ) //only one node in list
            {
                head = '\0';
                delete temp;
                return;
            }
            else
            {
                while( temp1->next != head )
                {
                    temp1 = temp1->next;
                }
                head = temp->next;
                temp1->next = head;
                delete temp;
            }
        }
        else
        {
            cout<<"Sorry!!! cList is empty."<<endl;
        }
    }

    void cList::midDelete(int dataLocation)
    {
        if(!isEmpty())
        {
            Node *prev;
            Node *temp;
            prev = 0;
            temp = head;

            for( int i=1; i<dataLocation; i++ )
            {
                prev = temp;
                temp = temp->next;

                if( temp == '\0' )
                {
                        cout<<"Sorry!!! No Node found with Data Value = "<<dataLocation<<"."<<endl;
                        return;
                }
            }

            if( prev == 0 )
            {
                this->beginDelete();
                return;
            }

            prev->next = temp->next;
            delete temp;
        }
        else
        {
            cout<<"Sorry!!! cList is empty."<<endl;
        }
    }

    void cList::endDelete()
    {
        if(!isEmpty())
        {
            Node *temp = head;
            Node *t;

            if( head->next == head )
            {
                delete temp;
                head = '\0';
            }
            else
            {
                t = head->next;
                while( t->next != head )
                {
                    t = t->next;
                    temp = temp->next;
                }
                temp->next = head;
                delete t;
            }
        }
        else
        {
            cout<<"Sorry!!! cList is Empty."<<endl;
        }
    }

    void cList::displaycList()
    {
        if(!isEmpty())
        {
            Node *temp = head;

            do
            {
                cout<<temp->data<<"  ";
                temp = temp->next;
            }
            while(temp != head);
            cout<<endl;
        }
        else
        {
            cout<<"Sorry!!! cList is Empty."<<endl;
        }
    }
```

Main.cpp File:

```
#include "cList.h"
#include "conio.h"
#include "Node.h"
#include <iostream>
using namespace std;

void main()
{
    cList l;

    cout<<"LINKED LIST EMPTY CEHCK RESULT:"<<endl;
    cout<<"==============================="<<endl;
    if(l.isEmpty())
    {
        cout<<"Yes. List is Empty."<<endl;
    }
    else
    {
        cout<<"No. List is not Empty"<<endl;
    }
    cout<<endl;

    cout<<"DISPLAY AFTER BEGIN NODE INSERTION:"<<endl;
    cout<<"==================================="<<endl;
    l.beginInsert(1);
    l.beginInsert(2);
    l.displaycList();
    cout<<endl;

    cout<<"DISPLAY AFTER END NODE INSERTION:"<<endl;
    cout<<"================================="<<endl;
    l.endInsert(3);
    l.endInsert(4);
    l.displaycList();
    cout<<endl;

    cout<<"DISPLAY AFTER-COUNTING NODES-NODE INSERTION:"<<endl; //count nodes and insert Node next to count == Z
    cout<<"============================================"<<endl;
    l.midInsert(90, 3);
    l.displaycList();
    cout<<endl;

    cout<<"DISPLAY AFTER-BEGIN NODE-DELETION:"<<endl;
    cout<<"=================================="<<endl;
    l.beginDelete();
    l.displaycList();
    cout<<endl;

    cout<<"DISPLAY AFTER-MID NODE-DELETION:"<<endl; //delete node with Position == Z
    cout<<"================================"<<endl;
    l.midDelete(3);
    l.displaycList();
    cout<<endl;

    cout<<"DISPLAY AFTER-END NODE-DELETION:"<<endl;
    cout<<"================================"<<endl;
    l.endDelete();
    l.displaycList();
    cout<<endl;

    getch();
}
```

Output:


C:\Users\Muhammad Anas Baig\Documents\Visual Studio 2010\Projects\Lab8-Ex1\Debug\Lab8-Ex...
11:49 PM
11-Apr-17

```
LINKED LIST EMPTY CEHCK RESULT:
===============================
Yes. List is Empty.

DISPLAY AFTER BEGIN NODE INSERTION:
===================================
2  1

DISPLAY AFTER END NODE INSERTION:
=================================
2  1  3  4

DISPLAY AFTER-COUNTING NODES-NODE INSERTION:
============================================
2  1  3  90  4

DISPLAY AFTER-BEGIN NODE-DELETION:
==================================
1  3  90  4

DISPLAY AFTER-MID NODE-DELETION:
================================
1  3  4

DISPLAY AFTER-END NODE-DELETION:
================================
1  3
```

Exercise 2 – Part 1:

Write C++ Function to Display the contents of alternate nodes of circular doubly linked list.

Solution:

Node.h File:

```
#pragma once
class Node
{
public:
    Node *prev;
    int data;
    Node *next;
public:
    Node(void);
};
```

Node.cpp File:

```
#include "Node.h"

Node::Node(void)
{
}
```

circularDoublyList.h File:

```
#include "Node.h"

#pragma once
class circularDoublyList
{
public:
    Node *head;
public:
    circularDoublyList(void);
    bool isEmpty();
    void insertBegin(int);
    void display();
    void displayAlternate();
};
```

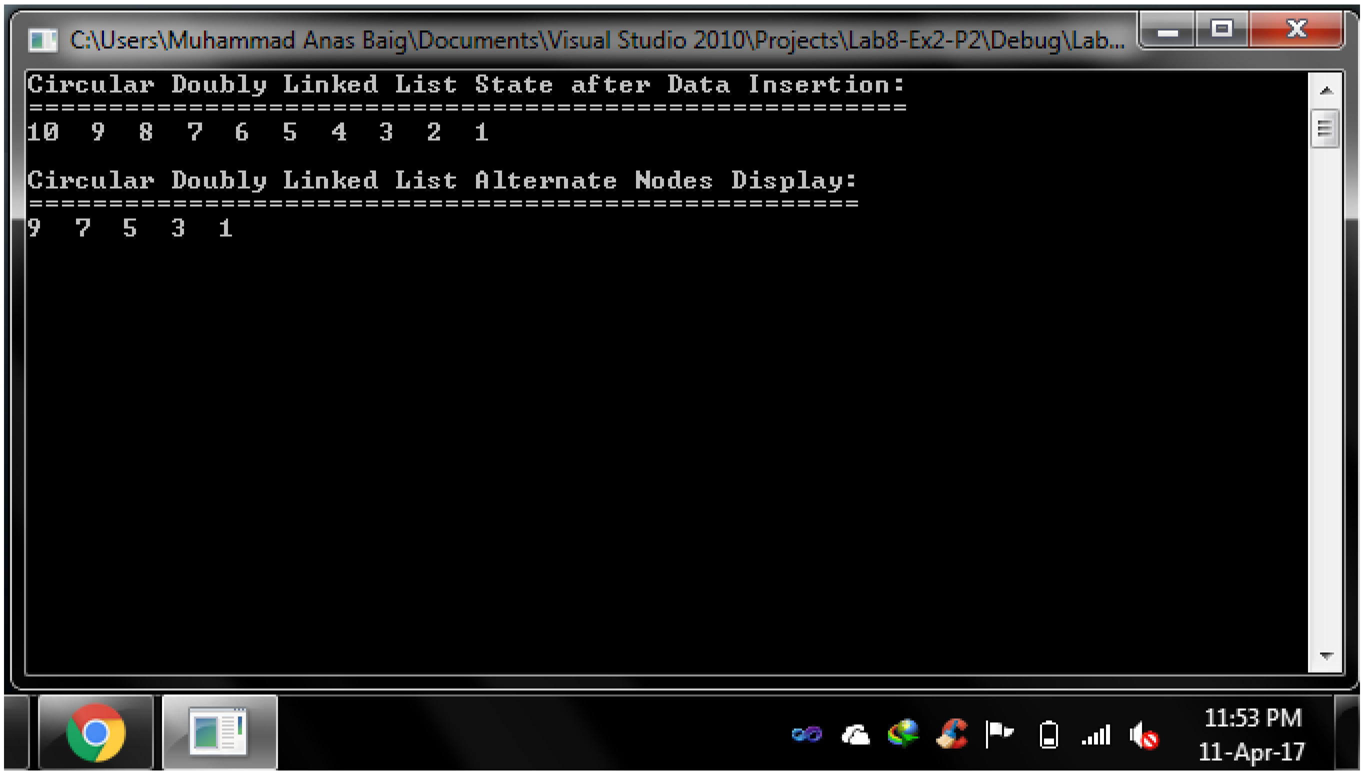
circularDoublyList.cpp File:

```
#include "circularDoublyList.h"
#include "Node.h"
#include <iostream>
using namespace std;

circularDoublyList::circularDoublyList(void)
{
    head = '\0';
}

bool circularDoublyList::isEmpty()
{
    if( head == '\0' )
    {
        return true;
    }
    else
    {
        return false;
    }
}

void circularDoublyList::insertBegin(int newVal)
{
    Node *ptr = new Node;
    ptr->prev = '\0';
    ptr->data = 0;
    ptr->next = '\0';

    if( !isEmpty() )
    {
        ptr->data = newVal;
        ptr->next = head;
        head->prev = ptr;
        head = ptr;
    }
    else
    {
        ptr->data = newVal;
        head = ptr;
    }
}

void circularDoublyList::display()
{
    if( !isEmpty() )
    {
        Node *temp = head;

        while( temp != '\0' )
        {
            cout<<temp->data<<"  ";
            temp = temp->next;
        }
        cout<<endl;
    }
    else
    {
        cout<<"SORRY!!! circularDoublyList is Empty."<<endl;
    }
}

void circularDoublyList::displayAlternate()
{
    if( !isEmpty() )
    {
        Node *temp = head;
        int count = 0;

        while( temp != '\0' )
        {
            count++;
            if(count%2 == 0)
            {
                cout<<temp->data<<"  ";
            }
            temp = temp->next;
        }
        cout<<endl;
    }
    else
    {
        cout<<"SORRY!!! circularDoublyList is Empty."<<endl;
    }
}
```

Main.cpp File:

```
#include "circularDoublyList.h"
#include "Node.h"
#include "conio.h"
#include <iostream>
using namespace std;

void main()
{
    circularDoublyList c;

    c.insertBegin(1);
    c.insertBegin(2);
    c.insertBegin(3);
    c.insertBegin(4);
    c.insertBegin(5);
    c.insertBegin(6);
    c.insertBegin(7);
    c.insertBegin(8);
    c.insertBegin(9);
    c.insertBegin(10);

    cout<<"Circular Doubly Linked List State after Data Insertion:"<<endl;
    cout<<"======================================================="<<endl;
    c.display();
    cout<<endl;

    cout<<"Circular Doubly Linked List Alternate Nodes Display:"<<endl;
    cout<<"===================================================="<<endl;
    c.displayAlternate();

    getch();
}
```

Output:



```
Circular Doubly Linked List State after Data Insertion:
=======================================================
10  9  8  7  6  5  4  3  2  1

Circular Doubly Linked List Alternate Nodes Display:
====================================================
9  7  5  3  1
```

C:\Users\Muhammad Anas Baig\Documents\Visual Studio 2010\Projects\Lab8-Ex2-P2\Debug\Lab...
11:53 PM
11-Apr-17

Exercise 2 – Part 2:

Write C++ Function to Reverse a singly linked list with dummy header node using stack (rearrange links not just data).

Solution:

Node.h File:

```cpp
#pragma once
class _Node
{
public:
    int data;
    _Node *prev;
    _Node *add;
    _Node *next;
public:
    _Node(void);
};
```

Node.cpp File:

```cpp
#include "_Node.h"

_Node::_Node(void)
{
}
```

Stack.h File:

```cpp
#include "_Node.h"

#pragma once
class doublyListStack
{
public:
    _Node *top;
public:
    doublyListStack(void);
    bool isEmpty();
    void push( _Node * );
    _Node * pop();
};
```

Stack.cpp File:

```cpp
#include "doublyListStack.h"
#include "_Node.h"
#include <iostream>
using namespace std;

doublyListStack::doublyListStack(void)
{
    top = '\0';
}

bool doublyListStack::isEmpty()
{
    if( top == '\0' )
    {
        return true;
    }
    else
    {
        return false;
    }
}

void doublyListStack::push(_Node * newVal)
{
    _Node *ptr = new _Node;
    ptr->prev = '\0';
    ptr->add = '\0';
    ptr->next = '\0';

    if( !isEmpty() )
    {
        ptr->add = newVal;
        ptr->next = top;
        top->prev = ptr;
        top = ptr;
    }
    else
    {
        ptr->add = newVal;
        top = ptr;
    }
}

_Node * doublyListStack::pop()
{
    if( !isEmpty() )
    {
        _Node *temp = top;
        _Node *tempadd = top->add;

        if( top->next == '\0' )
        {
            top = '\0';
        }
        else
        {
            top = top->next;
            top->prev = '\0';
        }
        return (tempadd);
    }
    else
    {
        cout<<"SORRY!!! List is Empty."<<endl;
        return ('\0');
    }
}
```

List.h File:

```
#include "_Node.h"
#include "_Node.h"

#pragma once
class list
{
public:
    _Node *head;
public:
    list(void);
    void add_Node(int);
    void display();
    void reverse();
};
```

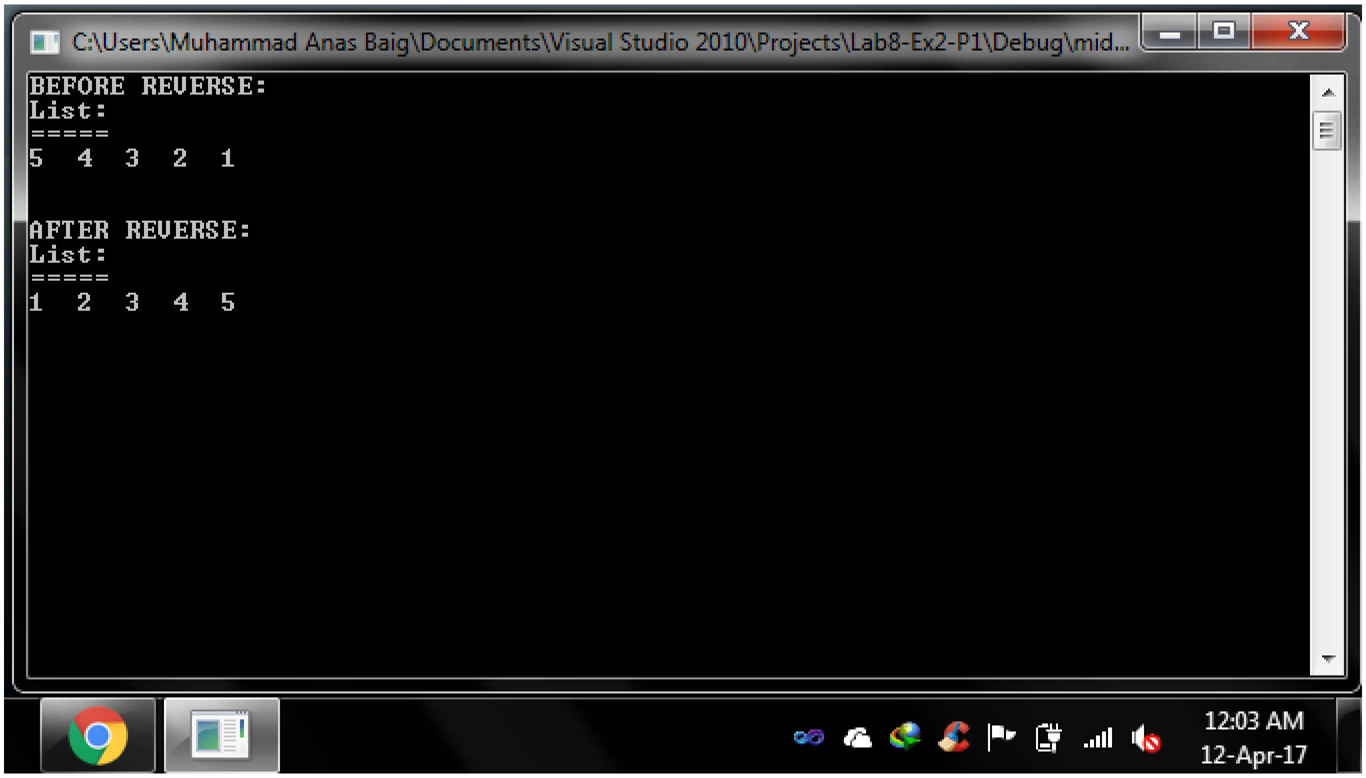
List.cpp File:

```
#include "list.h"
#include "_Node.h"
#include "doublyListStack.h"
#include <iostream>
using namespace std;

list::list(void)
{
    head = new _Node;
    head->next = '\0';
}

void list::add_Node(int data)
{
        _Node *ptr = new _Node;
        ptr->next = '\0';
        ptr->data = 0;

        ptr->data = data;

        ptr->next = head->next;
        head->next = ptr;
}

void list::display()
{
    if(head->next != '\0')
    {
        _Node *temp = head;

        cout<<"List:"<<endl;
        cout<<"====="<<endl;
        while(temp->next != '\0')
        {
            temp = temp->next;
            cout<<temp->data<<"  ";
        }
        cout<<endl;
    }
    else
    {
        cout<<"List is Empty."<<endl;
    }
}

void list::reverse()
{
    _Node *temp = head->next;
    doublyListStack d;

    while( temp != '\0' )
    {
        d.push( temp);
        temp = temp->next;
    }

    head->next = d.pop();
    temp = head->next;

    while( !d.isEmpty() )
    {
        temp->next = d.pop();
        temp = temp->next;
    }
    temp->next = '\0';
}
```

Main.cpp File:

```
#include "list.h"
#include "doublyListStack.h"
#include "_Node.h"
#include "conio.h"
#include <iostream>
using namespace std;

void main()
{
    list l;
    l.add_Node(1);
    l.add_Node(2);
    l.add_Node(3);
    l.add_Node(4);
    l.add_Node(5);

    cout<<"BEFORE REVERSE:"<<endl;
    l.display();
    cout<<endl;
    cout<<endl;

    l.reverse();

    cout<<"AFTER REVERSE:"<<endl;
    l.display();
    cout<<endl;


    getch();
}
```

Output:


C:\Users\Muhammad Anas Baig\Documents\Visual Studio 2010\Projects\Lab8-Ex2-P1\Debug\mid...
12:03 AM
12-Apr-17

```
BEFORE REVERSE:
List:
=====
5   4   3   2   1


AFTER REVERSE:
List:
=====
1   2   3   4   5
```