Task No 01:

Write a program to create a linked list and perform:

1. Insertion
2. Deletion

Solution:

Node (Class):

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DSA\_LAB\_03

{

internal class DoubleListNode

{

public int data;

public DoubleListNode previous;

public DoubleListNode next;

public DoubleListNode(int incomingData) {

data = incomingData;

previous = null;

next = null;

}

}

}

Double LinkedList (Class):

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace DSA\_LAB\_03

{

internal class DoubleLinkedList

{

public DoubleListNode head;

public DoubleLinkedList() {

head = null;

}

public void printForwardList()

{

DoubleListNode current = head;

while (current != null)

{

Console.Write(current.data+",");

current = current.next;

}

}

public void printBackwardList()

{

DoubleListNode current = head;

DoubleListNode last = null;

while (current != null)

{

last= current;

current = current.next;

}

while (last != null)

{

Console.Write(last.data+",");

last = last.previous;

}

}

public void push(int data)

{

DoubleListNode new\_node = new DoubleListNode(data);

if (head != null)

{

new\_node.next = head;

head.previous = new\_node;

}

head = new\_node;

}

public void insertAfter(int data,DoubleListNode prev)

{

if (prev == null)

{

Console.WriteLine("The given previous node is null !");

return;

}

DoubleListNode new\_node = new DoubleListNode(data);

if (prev.next != null)

{

new\_node.next = prev.next;

prev.next.previous = new\_node;

}

prev.next = new\_node;

new\_node.previous = prev;

}

public void insertBefore(int data, DoubleListNode after)

{

if (after == null)

{

Console.WriteLine("The specified Next node cant be null");

return;

}

if (after == head)

{

push(data);

return;

}

DoubleListNode new\_node = new DoubleListNode(data);

after.previous.next = new\_node;

new\_node.previous = after.previous;

new\_node.next = after;

after.previous = new\_node;

}

public void append(int data)

{

DoubleListNode current = head;

DoubleListNode last = null;

while (current != null)

{

last = current;

current = current.next;

}

DoubleListNode new\_node = new DoubleListNode(data);

last.next = new\_node;

new\_node.previous = last;

}

public void deleteNode(DoubleListNode deletionNode)

{

if (head == null || deletionNode == null)

{

Console.WriteLine("The specified node is not present in the list!");

return;

}

if (deletionNode == head)

{

head = head.next;

}

if (deletionNode.next != null)

{

deletionNode.next.previous = deletionNode.previous;

}

if (deletionNode.previous != null)

{

deletionNode.previous.next = deletionNode.next;

}

return;

}

public void deleteFromPosition(int n)

{

DoubleListNode current = head;

DoubleListNode deletionNode = null;

int i = 0;

while (current != null && i < n)

{

deletionNode= current;

current = current.next;

i++;

}

deleteNode(deletionNode);

}

}

}

(Insertion Methods Main Method Code)

(Insertion At Front):

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Runtime.Remoting.Messaging;

using System.Text;

using System.Threading.Tasks;

namespace DSA\_LAB\_03

{

internal class Program

{

static void Main(string[] args)

{

DoubleLinkedList my\_list = new DoubleLinkedList();

my\_list.push(1);

my\_list.push(5);

my\_list.push(4);

my\_list.push(6);

my\_list.printForwardList();

}

}

}

Output:

A black and white text

Description automatically generated

(Insertion After Specified Node):

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Runtime.Remoting.Messaging;

using System.Text;

using System.Threading.Tasks;

namespace DSA\_LAB\_03

{

internal class Program

{

static void Main(string[] args)

{

DoubleLinkedList my\_list = new DoubleLinkedList();

my\_list.head = new DoubleListNode(5);

DoubleListNode first = new DoubleListNode(8);

DoubleListNode second = new DoubleListNode(9);

DoubleListNode third = new DoubleListNode(5);

DoubleListNode fourth = new DoubleListNode(2);

DoubleListNode fifth = new DoubleListNode(10);

my\_list.head.next = first;

first.next = second;

first.previous = my\_list.head;

second.next = third;

second.previous = first;

third.next = fourth;

third.previous = second;

fourth.next = fifth;

fourth.previous = third;

fifth.previous = fourth;

my\_list.insertAfter(22, third);

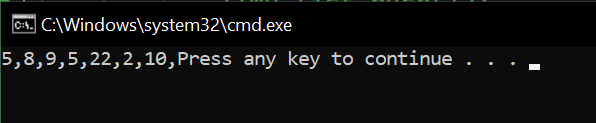
my\_list.printForwardList();

}

}

}

Output:



(Insertion Before Specified Node ):

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Runtime.Remoting.Messaging;

using System.Text;

using System.Threading.Tasks;

namespace DSA\_LAB\_03

{

internal class Program

{

static void Main(string[] args)

{

DoubleLinkedList my\_list = new DoubleLinkedList();

my\_list.head = new DoubleListNode(5);

DoubleListNode first = new DoubleListNode(8);

DoubleListNode second = new DoubleListNode(9);

DoubleListNode third = new DoubleListNode(5);

DoubleListNode fourth = new DoubleListNode(2);

DoubleListNode fifth = new DoubleListNode(10);

my\_list.head.next = first;

first.next = second;

first.previous = my\_list.head;

second.next = third;

second.previous = first;

third.next = fourth;

third.previous = second;

fourth.next = fifth;

fourth.previous = third;

fifth.previous = fourth;

my\_list.insertBefore(22, first);

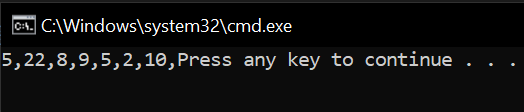
my\_list.printForwardList();

}

}

}

Output:



(Insertion At End):

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Runtime.Remoting.Messaging;

using System.Text;

using System.Threading.Tasks;

namespace DSA\_LAB\_03

{

internal class Program

{

static void Main(string[] args)

{

DoubleLinkedList my\_list = new DoubleLinkedList();

my\_list.head = new DoubleListNode(5);

DoubleListNode first = new DoubleListNode(8);

DoubleListNode second = new DoubleListNode(9);

DoubleListNode third = new DoubleListNode(5);

DoubleListNode fourth = new DoubleListNode(2);

DoubleListNode fifth = new DoubleListNode(10);

my\_list.head.next = first;

first.next = second;

first.previous = my\_list.head;

second.next = third;

second.previous = first;

third.next = fourth;

third.previous = second;

fourth.next = fifth;

fourth.previous = third;

fifth.previous = fourth;

my\_list.append(47);

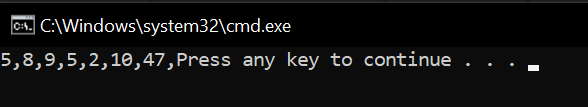
my\_list.printForwardList();

}

}

}

Output:



(Deletion Cases)

(Specified Node Deletion):

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Runtime.Remoting.Messaging;

using System.Text;

using System.Threading.Tasks;

namespace DSA\_LAB\_03

{

internal class Program

{

static void Main(string[] args)

{

DoubleLinkedList my\_list = new DoubleLinkedList();

my\_list.head = new DoubleListNode(5);

DoubleListNode first = new DoubleListNode(8);

DoubleListNode second = new DoubleListNode(9);

DoubleListNode third = new DoubleListNode(5);

DoubleListNode fourth = new DoubleListNode(2);

DoubleListNode fifth = new DoubleListNode(10);

my\_list.head.next = first;

first.next = second;

first.previous = my\_list.head;

second.next = third;

second.previous = first;

third.next = fourth;

third.previous = second;

fourth.next = fifth;

fourth.previous = third;

fifth.previous = fourth;

my\_list.deleteNode(fourth);

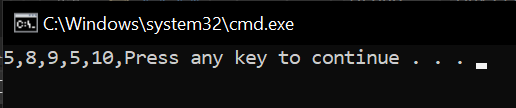
my\_list.printForwardList();

}

}

}

Output:



(Specified Position Node Deletion):

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Runtime.Remoting.Messaging;

using System.Text;

using System.Threading.Tasks;

namespace DSA\_LAB\_03

{

internal class Program

{

static void Main(string[] args)

{

DoubleLinkedList my\_list = new DoubleLinkedList();

my\_list.head = new DoubleListNode(5);

DoubleListNode first = new DoubleListNode(8);

DoubleListNode second = new DoubleListNode(9);

DoubleListNode third = new DoubleListNode(5);

DoubleListNode fourth = new DoubleListNode(2);

DoubleListNode fifth = new DoubleListNode(10);

my\_list.head.next = first;

first.next = second;

first.previous = my\_list.head;

second.next = third;

second.previous = first;

third.next = fourth;

third.previous = second;

fourth.next = fifth;

fourth.previous = third;

fifth.previous = fourth;

my\_list.deleteFromPosition(1);

my\_list.printForwardList();

}

}

}

Output:

