INDEX

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| SNO | DATE | LAB NO | LAB OBJECTIVE | SIGN |
| 01 | 14-10-21 | 01 | Arrays |  |
| 02 | 21-10-21 | 02 | Linear Search & Sorting |  |
| 03 | 28-10-21 | 03 | Linked List |  |
| 04 | 4-11-21 | 04 | Doubly & Circular Linked List |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

Bahria University,

Karachi Campus



LAB EXPERIMENT NO.

**03**

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
| 01 | **Write a program to create a linked list and perform**  **Traversing Insertion deletion** |

Submitted On:

**11-11-2021**

(Date: DD/MM/YY)

Task 1: **Write a program to create a linked list and perform**

**Traversing  
Insertion  
deletion**

Solution:

public class Node

{

public int data;

public Node next;

public Node(int d)

{

data = d;

next = null;

} // Constructor

}

public class LinkedList

{

public Node head;

// head of list

public void push(int new\_data)

{

Node new\_node = new Node(new\_data);

new\_node.next = head;

head = new\_node;

}

public void insertAfter(Node prev\_node, int new\_data)

{

if (prev\_node == null)

{

Console.WriteLine("Previous Node Cannot Be Null...!!!");

return;

}

Node new\_node = new Node(new\_data);

new\_node.next = prev\_node.next;

prev\_node.next = new\_node;

}

public void append(int new\_data)

{

Node new\_node = new Node(new\_data);

if (head == null)

{

head = new Node(new\_data);

return;

}

new\_node.next = null;

Node last = head;

while (last.next != null)

last = last.next;

last.next = new\_node;

return;

}

public void deleteNode(int key)

{

Node temp = head, prev = null;

if (temp != null &&

temp.data == key)

{

head = temp.next;

return;

}

while (temp != null &&

temp.data != key)

{

prev = temp;

temp = temp.next;

}

if (temp == null)

return;

prev.next = temp.next;

}

public void printList()

{

Node n = head;

Console.WriteLine();

while (n != null)

{

Console.Write(" "+ n.data);

n = n.next;

}

}

}

class Program

{

static void Main(string[] args)

{

LinkedList llist = new LinkedList();

llist.head = new Node(10);

Node second = new Node(20);

Node third = new Node(30);

llist.head.next = second; // Link first node with the second node

second.next = third;

Console.WriteLine("\nInitial Linked List");

llist.printList();

llist.push(9);

Console.WriteLine("\n\nAdd Node At Front Of The Linked List");

llist.printList();

llist.insertAfter(second, 12);

Console.WriteLine("\n\nAdd Node After A Given Node");

llist.printList();

llist.append(32);

Console.WriteLine("\n\nAdd Node At End Of The Linked List");

llist.printList();

llist.deleteNode(10);

Console.WriteLine("\n\nDeletion");

llist.printList();

Console.ReadLine();

}

**Output:**

Text

Description automatically generated

Bahria University,

Karachi Campus

A picture containing text, room

Description automatically generated

LAB EXPERIMENT NO.

**04**

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
| 01 | **Create a doubly linked list and apply following:**  **Insertion at beginning**  **Insertion at a given node (before and after both)**  **Delete node at any point. Delete node at end**  **Traversal** |

Submitted On:

**11-11-2021**

(Date: DD/MM/YY)

Task 1: **Create a doubly linked list and apply following:**

**Insertion at beginning**

**Insertion at a given node (before and after both)**

**Delete node at any point.  
Delete node at end**

**Traversal**

Solution:

public class Node

{

public int data;

public Node prev;

public Node next;

public Node(int d) { data = d; }

}

public class DLL

{

public Node head;

public void printlist(Node node)

{

Node last = null;

Console.WriteLine("\nTraversing in forward Direction");

Console.WriteLine();

while (node != null)

{

Console.Write(node.data + " ");

last = node;

node = node.next;

}

Console.WriteLine();

Console.WriteLine("\nTraversing in backward direction");

Console.WriteLine();

while (last != null)

{

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

last = last.prev;

}

Console.WriteLine("\n");

}

public void push(int new\_data)

{

Node new\_Node = new Node(new\_data);

new\_Node.next = head;

new\_Node.prev = null;

if (head != null)

head.prev = new\_Node;

head = new\_Node;

}

public void InsertBefore(Node next\_Node, int new\_data)

{

Node new\_node = new Node(new\_data);

if (next\_Node == null)

{

Console.WriteLine("The given next node cannot be NULL ");

return;

}

new\_node.prev = next\_Node.prev;

next\_Node.prev = new\_node;

new\_node.next = next\_Node;

if (new\_node.next != null)

new\_node.prev.next = new\_node;

else if (new\_node.prev == null)

head = new\_node;

}

public void append(int new\_data)

{

Node new\_node = new Node(new\_data);

Node last = head; /\* used in step 5\*/

new\_node.next = null;

if (head == null)

{

new\_node.prev = null;

head = new\_node;

return;

}

while (last.next != null)

last = last.next;

last.next = new\_node;

new\_node.prev = last;

}

public void InsertAfter(Node prev\_Node, int new\_data)

{

if (prev\_Node == null)

{

Console.WriteLine("The given previous node cannot be NULL ");

return;

}

Node new\_node = new Node(new\_data);

new\_node.next = prev\_Node.next;

prev\_Node.next = new\_node;

new\_node.prev = prev\_Node;

if (new\_node.next != null)

new\_node.next.prev = new\_node;

}

public void deleteNode(Node del)

{

if (head == null || del == null)

{

return;

}

if (head == del)

{

head = del.next;

}

if (del.next != null)

{

del.next.prev = del.prev;

}

if (del.prev != null)

{

del.prev.next = del.next;

}

return;

}

}

class Program

{

static void Main(string[] args)

{

DLL list = new DLL();

list.head = new Node(10);

Node second = new Node(20);

Node third = new Node(30);

list.head.next = second;

second.next = third;

second.prev = list.head;

third.prev = second;

Console.WriteLine("Initial Doubly Linked List");

list.printlist(list.head);

Console.WriteLine("Inserton Before Node");

list.InsertBefore(second, 5);

list.printlist(list.head);

Console.WriteLine("Insertion After Node");

list.InsertAfter(list.head, 9);

list.printlist(list.head);

Console.WriteLine("Push In Doubly Linked List");

list.push(1);

list.printlist(list.head);

Console.WriteLine("Appending In Doubly Linked List");

list.append(50);

list.printlist(list.head);

Console.WriteLine("Deleting In Doubly Linked List");

list.deleteNode(list.head);

list.printlist(list.head);

Console.ReadLine();

**Output:**

Text

Description automatically generated