Bahria University,

Karachi Campus

A picture containing text, room

Description automatically generated

LAB EXPERIMENT NO.

**03**

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
|  | **Linear Search & Sorting Algorithms** |
| 1 | Write a program to create a linked list and pertform  \*traversing  \* Insertion  \*deletion |

Submitted On:

(Date: DD/MM/YY)

**TASK # 01:** **Write a program to create a linked list and pertform**

**\*traversing**

**\* Insertion**

**\*deletion**

**SOLUTION:**

public class LinkedList

{

public Node head;

public void printList()

{

Node n = head;

while (n != null)

{

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

n = n.next;

}

}

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("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;

}

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;

}

}

**static void Main(string[] args)**

{

LinkedList llist = new LinkedList();

llist.head = new Node(1);

Node second = new Node(2);

Node third = new Node(3);

llist.head.next = second;

second.next = third;

Console.WriteLine("LINKLIST ELEMENTS :");

llist.printList();

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

char ch;

do

{

Console.WriteLine("ENTER OPTION WHAT YOU WANT TO DO :");

Console.WriteLine("1) Traversing");

Console.WriteLine("2) Insertion");

Console.WriteLine("3) Deletion");

int answer = int.Parse(Console.ReadLine());

if (answer==1)

{

Console.WriteLine("Simple linlist items print: ");

llist.printList();

Console.WriteLine();

}

else if (answer==2)

{

do

{

Console.WriteLine("Where You Insert Element :");

Console.WriteLine("1) Beggining : ");

Console.WriteLine("2) Middle : ");

Console.WriteLine("3) Last : ");

int replay = int.Parse(Console.ReadLine());

switch (replay)

{

case 1:

Console.WriteLine("Add Elements at Beggining :");

llist.push(5);

llist.printList();

Console.WriteLine();

break;

case 2:

Console.WriteLine("Add Elements at middle:");

llist.insertAfter(second, 6);

llist.printList();

Console.WriteLine();

break;

case 3:

Console.WriteLine("Elements Add at last :");

llist.append(90);

llist.printList();

break;

default:

Console.WriteLine("Invalid Option :");

break;

}

Console.WriteLine("IF YOU WANT TO DO AGAIN PERFORM INSERTION[Y/N]");

ch = Char.Parse(Console.ReadLine());

} while (ch=='y');

}

else if (answer==3)

{

Console.WriteLine("Delete element");

llist.deleteNode(1);

llist.printList();

Console.WriteLine();

}

else

{

Console.WriteLine("Invalid Option");

}

Console.WriteLine("IF YOU WANT TO DO AGAIN [Y/N]");

ch = Char.Parse(Console.ReadLine());

} while (ch=='y');

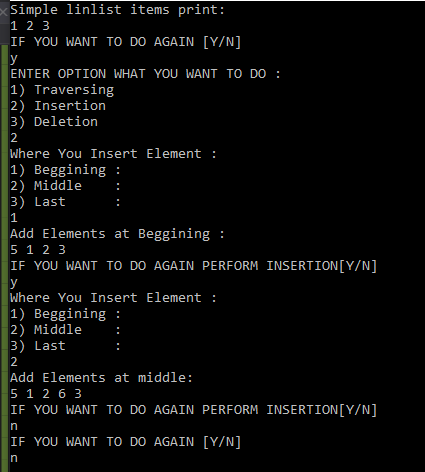
Console.ReadLine();

}

}

}

**OUTPUT:**

****