**Task No. 1:** Write a program to create a linked list and Perform.

1) Traversing

2) Insertion

3) Deletion

**Solution:**

using System;

namespace LAB\_3\_DSA

{

class node

{

public int data;

public node next;

public node(int a)

{

data = a;

next = null;

}

}

class LinkList

{

// Head is Used to Identify the Starting Of The Linklist

public node head;

// To Insert At the First Of The Linklist

public void PushElement(int new\_data)

{

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

new\_node.next = head;

head = new\_node;

}

// To Insert At the End Of The Linklist

public void appendElement(int new\_data)

{

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

if (head == null)

{

head = new node(new\_data);

return;

}

node last = head;

while (last.next != null)

{

last = last.next;

}

last.next = new\_node;

return;

}

//To Insert At Index The Given Index Of The Linklist

public void InsertAtIndex(int new\_data, int index)

{

node p = head;

int i = 0;

while (i != (index - 1))

{

p = p.next;

i++;

}

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

new\_node.next = p.next;

p.next = new\_node;

}

// To Insert After the Previous Node

public void InsertAfter(node previous\_node, int new\_data)

{

if (previous\_node == null)

{

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

return;

}

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

new\_node.next = previous\_node.next;

previous\_node.next = new\_node;

}

// To Print The Element Of The LinkList

public void print()

{

node n = head;

Console.WriteLine("Element Present in LinkList Are ");

while (n != null)

{

Console.WriteLine($" {n.data} ");

n = n.next;

}

Console.WriteLine("``````````````````````````````````````");

}

//------------------------------------------------------------------

// Delete Operation In Linklist

//------------------------------------------------------------------☻

//Delete The First Element In The LinkList

public void DeleteFirst()

{

//node temp = head;

//head = temp.next;

head = head.next;

}

//Delete The Last Element In The LinkList

public void DeleteLast()

{

node temp = head;

node q = head.next;

while (q.next != null)

{

temp = temp.next;

q = q.next;

}

temp.next = null;

}

public void DeleteAtIndex(int index)

{

node p = head;

node q = head.next;

int i = 0;

while (i != (index - 1))

{

p = p.next;

q = q.next;

i++;

}

p.next = q.next;

}

public void DeleteTheValue(int key)

{

node p = head;

node q = head.next;

int i = 0;

while (q.data != key && q.next != null)

{

p = p.next;

q = q.next;

i++;

}

if (q.data == key)

{

p.next = q.next;

}

return;

}

}

class Program

{

static void Main(string[] args)

{

byte res;

LinkList llist = new LinkList();

llist.head = new node(1);

node second = new node(2);

node third = new node(3);

llist.head.next = second;

second.next = third;

do

{

Console.WriteLine("Please Choose AN Option Below :");

Console.WriteLine(" 1 ) Traversing");

Console.WriteLine(" 2 ) Insertion");

Console.WriteLine(" 3 ) Deletion");

Console.Write("Enter : ");

res = byte.Parse(Console.ReadLine());

switch (res)

{

case 1:

Console.Clear();

llist.print();

break;

case 2:

Console.Clear();

Console.WriteLine("Please Choose AN Option Below :");

Console.WriteLine(" 1 ) Insert At Front");

Console.WriteLine(" 2 ) Insert At Middle");

Console.WriteLine(" 3 ) Insert At Given Index ");

Console.WriteLine(" 4 ) Insert At Last");

Console.Write("Enter : ");

byte res1 = byte.Parse(Console.ReadLine());

if (res1 == 1)

{

Console.Write("Enter Element You Want To Insert : ");

llist.PushElement(int.Parse(Console.ReadLine()));

}

else if (res1 == 2)

{

Console.Write("Enter Node Data : ");

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

//Console.Write("Enter Previous Node Name : ");

//string nodename = Console.ReadLine();

llist.InsertAfter(second, nodedata);

}

else if (res1 == 3)

{

int index;

Console.Write("Enter Element You Want To Insert : ");

int res12 = int.Parse(Console.ReadLine());

do

{

Console.Write("Enter Index You Want To Insert ( Greater Then \" 0 \" ) : ");

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

} while (index <= 0);

llist.InsertAtIndex(res12, index);

}

else if (res1 == 4)

{

Console.Write("Enter Element You Want To Insert : ");

llist.appendElement(int.Parse(Console.ReadLine()));

}

break;

case 3:

Console.WriteLine("Please Choose AN Option Below :");

Console.WriteLine(" 1 ) Delete At Front");

Console.WriteLine(" 2 ) Delete At Given Index ");

Console.WriteLine(" 3 ) Delete At Value");

Console.WriteLine(" 4 ) Delete At Last");

Console.Write("Enter : ");

byte delres = byte.Parse(Console.ReadLine());

if (delres == 1)

{

llist.DeleteFirst();

}

else if (delres == 2)

{

int index;

do

{

Console.Write("Enter Element You Want To Delete ( Greater Then \" 0 \" ) : ");

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

} while (index <= 0);

llist.DeleteAtIndex(index);

}

else if (delres == 3)

{

int value;

do

{

Console.Write("Enter Value You Want To Delete ( Greater Then \" 1 \" ) : ");

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

} while (value <= 1);

llist.DeleteTheValue(value);

}

else if (delres == 4)

{

llist.DeleteLast();

}

break;

default:

break;

}

} while (res != 1 || res != 2 || res != 3);

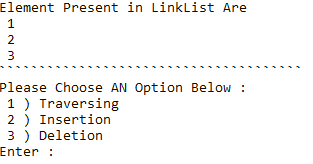
}

}

}

**Output:**

**A picture containing timeline

Description automatically generated**

A picture containing text

Description automatically generated**Text

Description automatically generatedText

Description automatically generated**