Bahria University,

Karachi Campus



LAB EXPERIMENT NO.

\_\_\_9\_\_\_\_

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
| 1 | Implement bucket sort using linked list. |
| 2 | Create static tree and perform in order, pre order and post order traversal. Also search a required node in the tree. |
|  |  |
|  |  |

Submitted On:

Date: 6/1/2022

**Task No. 1: Implement bucket sort using linked list.**

**Solution:**

**Class**

public class Node

{

public int data;

public Node next;

public Node(int d)

{

data = d;

next = null;

}

}

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 naya\_data)

{

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

nai\_node.next = head;

head = nai\_node;

}

public void append(int naya\_data)

{

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

if (head == null)

{

head = nai\_node;

return;

}

nai\_node.next = null;

Node last = head;

while (last.next != null)

{

last = last.next;

}

last.next = nai\_node;

return;

}

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

{

if (prev\_node == null)

Console.WriteLine("Given previous node cannot be null");

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

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

prev\_node.next = new\_node;

}

public void Delete(int data)

{

Node temp = head, prev = null;

if (temp != null && temp.data == data)

head = temp.next;

while (temp != null && temp.data != data)

{

prev = temp;

temp = temp.next;

}

if (temp == null)

return;

prev.next = temp.next;

}

public void BucketSort()

{

Node i = head;

while (i.next != null)

{

Node small = i;

Node j = i.next;

while (j.next != null)

{

if (i.data > j.data)

small = j;

j = j.next;

}

int t = i.data;

i.data = small.data;

small.data = i.data;

i = i.next;

}

}

}

**Main Method**

static void Main(string[] args)

{

LinkedList buk1 = new LinkedList();

LinkedList buk2 = new LinkedList();

LinkedList buk3 = new LinkedList();

Console.WriteLine("Enter no of values you want to enter:");

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

int[] arr = new int[n];

for(int i = 0; i < n; i++)

{

Console.WriteLine("Enter values");

arr[i] = int.Parse(Console.ReadLine());

}

Console.WriteLine("Before Sorting");

for(int i = 0; i < n; i++)

{

Console.Write(arr[i]+" ");

}

Console.WriteLine();

for(int i = 0; i < n; i++)

{

if (arr[i] >= 0 && arr[i] < 5)

{

buk1.append(arr[i]);

buk1.BucketSort();

}

else if (arr[i] >= 5 && arr[i] < 10)

{

buk2.append(arr[i]);

buk2.BucketSort();

}

else if (arr[i] > 10)

{

buk3.append(arr[i]);

buk3.BucketSort();

}

}

Console.WriteLine("After sorting");

buk1.printList();

buk2.printList();

buk3.printList();

Console.WriteLine();

}

}

Text

Description automatically generated**Output:**

**Task No. 2: Create static tree and perform in order, pre order and post order traversal. Also search a required node in the tree.**

**Solution:**

**Main method**

static void Main(string[] args)

{

Console.Write("How many Layers in tree ? ");

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

if (layer > 0)

{

Console.Write("Enter the value of root = ");

string root = Console.ReadLine();

Tree obj = new Tree(root, layer);

obj.inputTree();

obj.treeInArray();

obj.displayTree();

obj.preOrder();

obj.inOrder();

obj.postOrder();

obj.search();

}

else

{

Console.WriteLine("Incorrect");

}

}

**Output:**

