Bahria University,

Karachi Campus

A picture containing text, room

Description automatically generated

LAB EXPERIMENT NO.

**09**

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
|  | **BUCKET, RADIX SORT & BINARY TREES** |
| 1 | Implement bucket sort using linked list |
| 2 | Create static tree and perform inorder, preorder and post order traversal. Also search a required node in the tree |

Submitted On:

(Date: DD/MM/YY)

**Task 1: Implement bucket sort using linked list.**

**OUTPUT**:

**Text

Description automatically generated**

**Solution:**

public static void BucketSort(ref int[] data)

{

int min = data[0];

int max = data[0];

for (int i = 1; i < data.Length; i++)

{

if (data[i] > max)

max = data[i];

if (data[i] < min)

min = data[i];

}

List<int>[] bucket = new List<int>[max - min + 1];

for (int i = 0; i < bucket.Length; i++)

bucket[i] = new List<int>();

for (int i = 0; i < data.Length; i++)

bucket[data[i] - min].Add(data[i]);

int k = 0;

for (int i = 0; i < bucket.Length; i++)

{

if (bucket[i].Count > 0)

{

for (int j = 0; j < bucket[i].Count; j++)

{

data[k] = bucket[i][j];

k++;

}

}

}

}

public static void Display(int[] arr)

{

for (int i = 0; i < arr.Length; i++)

{

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

}

Console.WriteLine();

}

static void Main(string[] args)

{

int[] arr = { 3, 2, 7, 4, 11, 1, 5, 10 };

Console.WriteLine("Original Array: ");

Display(arr);

BucketSort(ref arr);

Console.WriteLine("After performing bucket sort: ");

Display(arr);

Console.ReadLine();

}

}

}

**TASK NO 2:** Create static tree and perform inorder, preorder and post order traversal. Also search a required node in the tree.

**SOLUTION:**

class Node

{

public int data;

public Node Left\_child;

public Node Right\_child;

public Node(int data)

{

this.data = data;

Left\_child = null;

Right\_child = null;

}

}

class BinaryTree

{

public void Inorder(Node node)

{

if (node != null)

{

Inorder(node.Left\_child);

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

Inorder(node.Right\_child);

} }

public void PreOrder(Node node)

{

if (node!=null)

{

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

PreOrder(node.Left\_child);

PreOrder(node.Right\_child);

}

}

public void PostOrder(Node node)

{

if (node!=null)

{

PostOrder(node.Left\_child);

PostOrder(node.Right\_child);

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

}

} }

class Program

{

static void Main(string[] args)

{

BinaryTree obj = new BinaryTree();

Node n1 = new Node(5);

Node n2 = new Node(10);

Node n3 = new Node(7);

Node n4 = new Node(2);

Node n5 = new Node(8);

n1.Left\_child = n2;

n1.Right\_child = n3;

n2.Left\_child = n4;

n2.Right\_child = n5;

Console.Write("Pre Order Traversal :");

obj.PreOrder(n1);

Console.Write("\n\nIn Order Traversal :");

obj.Inorder(n1);

Console.Write("\n\nPost Order Traversal :");

obj.PostOrder(n1);

Console.ReadLine();

}}}

**OUTPUT:**

Graphical user interface, text

Description automatically generated