Q: - Implement Radix and bucket sort.

Input:-

Bucket Sort:-

public List<int> BucketSort1(params int[] x)

{

List<int> result = new List<int>(); //save the sorted result of buckets

int numOfBuckets = 10;

List<int>[] bucket = new List<int>[numOfBuckets];

//assign list to each bucket

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

{

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

}

//use of formula to put element in a specific bucket

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

{

int bucketChoise = (x[i] / numOfBuckets);

bucket[bucketChoise].Add(x[i]); //e.g;means add 99 in bucket 9

Console.Write("Value {0} Bucket No. = {1}", x[i], bucketChoise);

Console.WriteLine();

}

//Sort each bucket one by one and add sorted result to the list

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

{

int[] temp = BubbleSortList(bucket[i]);

result.AddRange(temp);

}

return result;

}

//Sorting each bucket by Bubble Sort

public int[] BubbleSortList(List<int> input)

{

for (int i = 0; i < input.Count; i++)

{

for (int j = 0; j < input.Count; j++)

{

if (input[i] < input[j])

{

int temp = input[i];

input[i] = input[j];

input[j] = temp;

}

}

}

return input.ToArray();

}

Main:-

Console.WriteLine("Bucket Sort Code");

BucketSort bs = new BucketSort();

Console.WriteLine("Enter Length of Array?");

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

int[] x = new int[limit];

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

{

Console.Write("Enter value : ");

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

}

Console.WriteLine();

//int[] x = new int[] { 99,56,45,8,74,12,45,15,7,15,25,25,21,35,45,21,11,56,65,78,89,90 };

List<int> FinalResult = bs.BucketSort1(x);

Console.WriteLine("Sorted Array values are given below:");

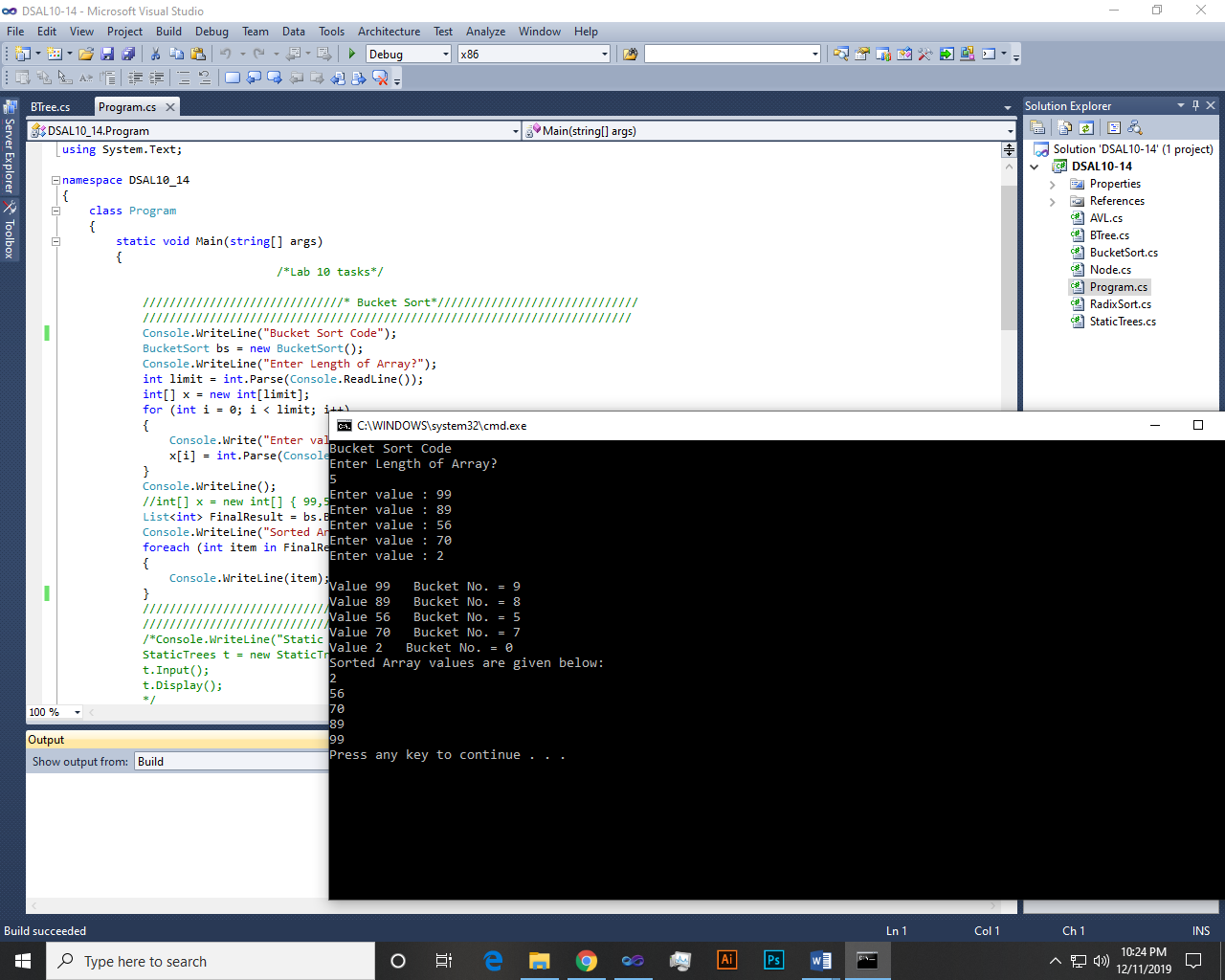
foreach (int item in FinalResult)

{

Console.WriteLine(item);

}

Output:-



Radix Sort:-

private int[] data;

private IList<IList<int>> digits = new List<IList<int>>();

private int maxLength = 0;

public RadixSort()

{

for (int i = 0; i < 10; i++)

{

digits.Add(new List<int>());

}

Console.Write("Enter the Length of an array : ");

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

data = new int[count];

Console.ReadLine();

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

{

Console.Write("Enter number {0} : ", i + 1);

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

if (maxLength < data[i].ToString().Length)

maxLength = data[i].ToString().Length;

}

}

public void Radix()

{

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

{

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

{

int digit = (int)((data[j] % Math.Pow(10, i + 1)) / Math.Pow(10, i));

digits[digit].Add(data[j]);

}

int index = 0;

for (int k = 0; k < digits.Count; k++)

{

IList<int> selDigit = digits[k];

for (int l = 0; l < selDigit.Count; l++)

{

data[index++] = selDigit[l];

}

}

ClearDigits();

}

printSortedData();

}

private void ClearDigits()

{

for (int k = 0; k < digits.Count; k++)

{

digits[k].Clear();

}

}

public void printSortedData()

{

Console.WriteLine("The Sorted Numbers are : ");

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

{

Console.WriteLine(data[i]);

}

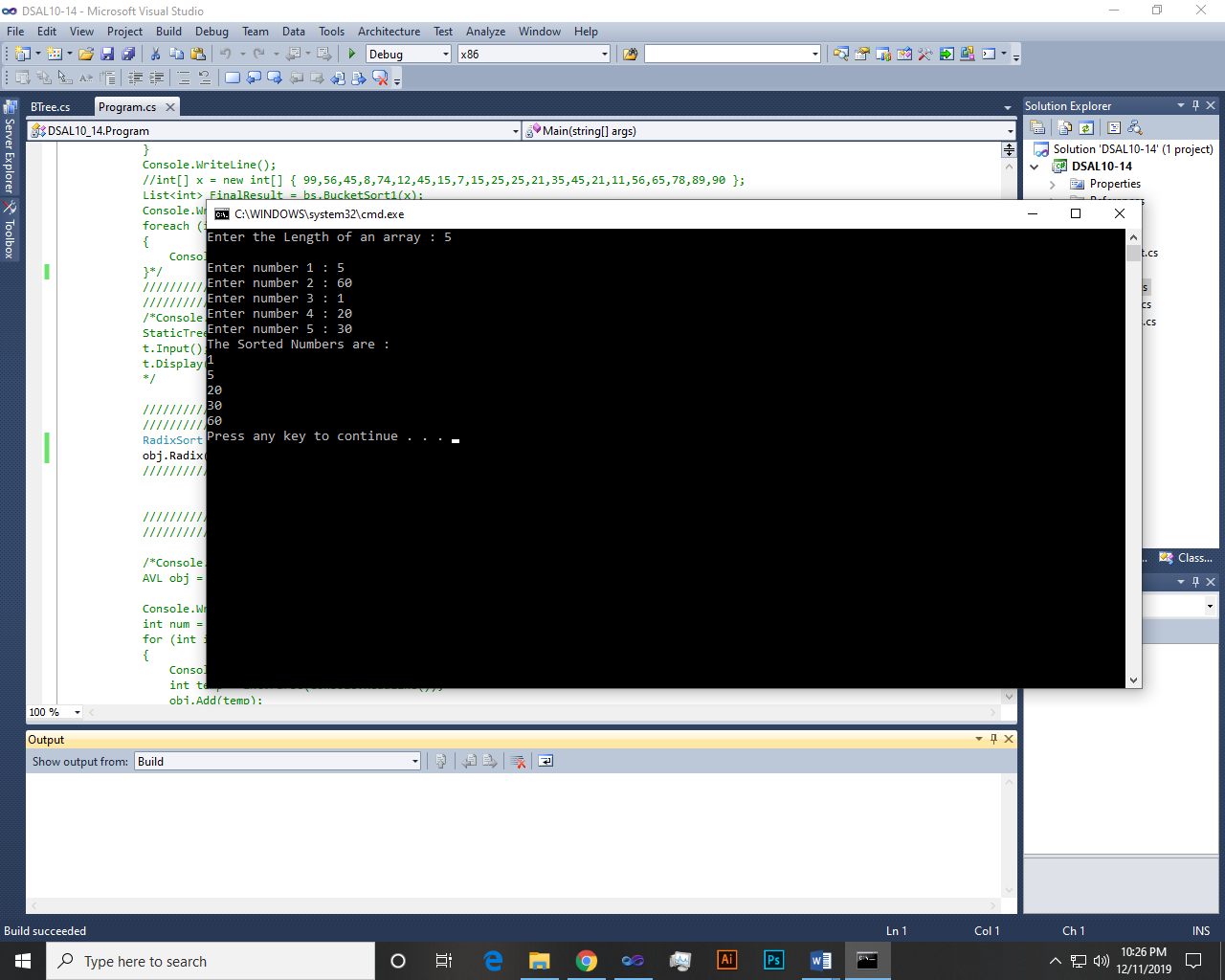
}

Main:-

RadixSort obj = new RadixSort();

obj.Radix();

Output:-



Q: - Create static tree and perform inorder, preorder and post order traversal. Also search a required node in the tree.

Input:-

Tree:-

class DynamicTree

{

internal Node1 root;

public DynamicTree()

{

root = null;

}

public void Postorder(Node1 node)

{

if (node == null)

return;

Postorder(node.left);

Postorder(node.right);

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

}

public void Inorder(Node1 node)

{

if (node == null)

return;

Inorder(node.left);

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

Inorder(node.right);

}

public void Preorder(Node1 node)

{

if (node == null)

return;

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

Preorder(node.left);

Preorder(node.right);

}

public void Postorder()

{

Postorder(root);

}

public void Inorder()

{

Inorder(root);

}

public void Preorder()

{

Preorder(root);

}

}

Node:-

public int key;

public Node1 left, right;

public Node1(int item)

{

key = item;

left = right = null;

}

Main:-

Console.WriteLine("Tree Traversals");

DynamicTree tree = new DynamicTree();

tree.root = new Node1(11);

tree.root.left = new Node1(12);

tree.root.right = new Node1(13);

tree.root.left.left = new Node1(14);

tree.root.left.right = new Node1(15);

Console.WriteLine("\_\_\_\_\_\_\_Preorder\_\_\_\_\_\_\_ ");

tree.Preorder();

Console.WriteLine("\n\_\_\_\_\_\_\_Inorder\_\_\_\_\_\_\_ ");

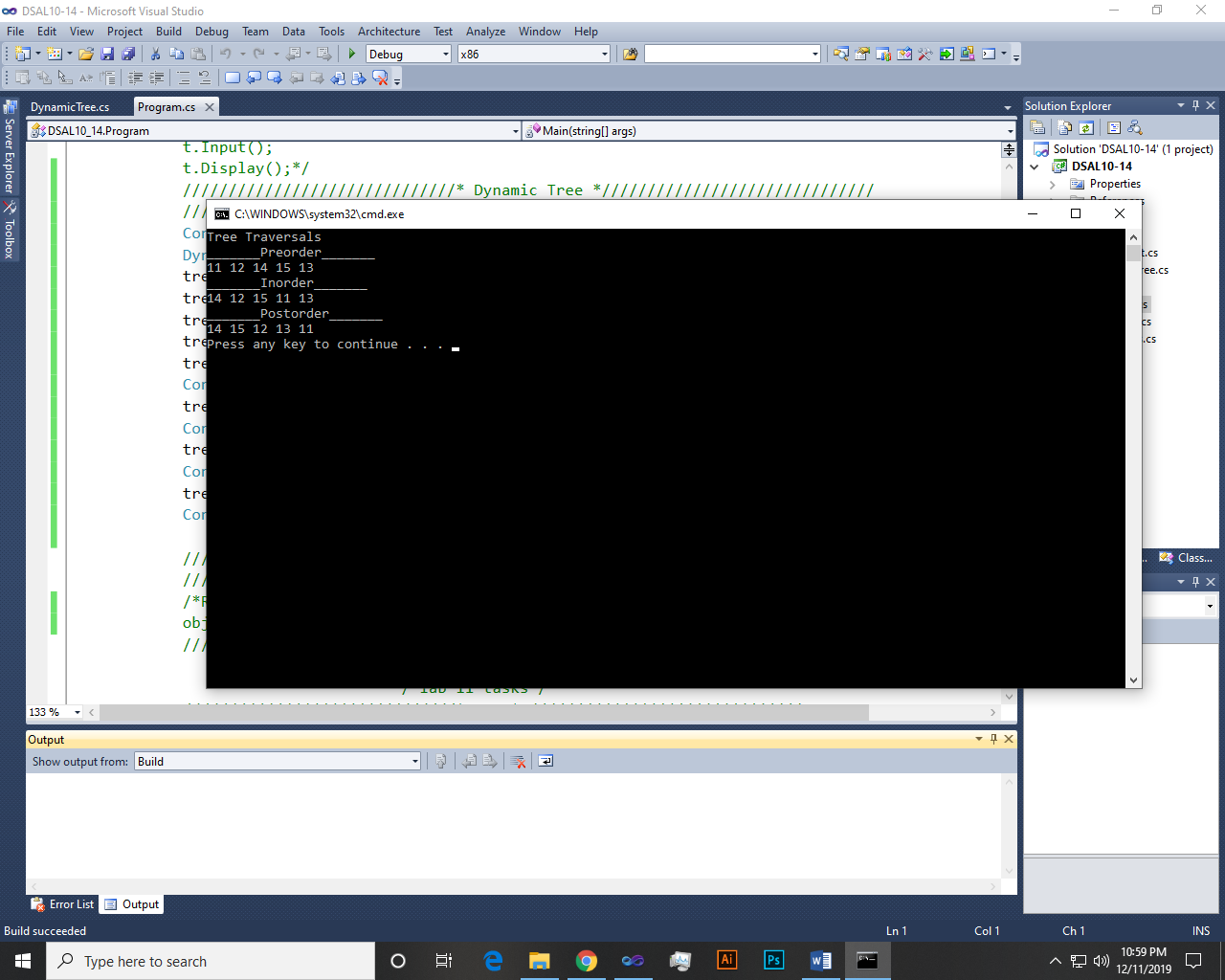
tree.Inorder();

Console.WriteLine("\n\_\_\_\_\_\_\_Postorder\_\_\_\_\_\_\_");

tree.Postorder();

Console.WriteLine();

Output:-



~~~~~~\*\*/**THE END**/\*\*~~~~~~