Bahria University,

Karachi Campus

A picture containing text, room

Description automatically generated

LAB EXPERIMENT NO.

\_\_\_\_**09**\_\_\_\_\_

LIST OF TASKS

|  |  |
| --- | --- |
| TASK NO | OBJECTIVE |
| 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

21/12/2022

(Date: DD/MM/YY)

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

**Solution:**

**Main Class:**

static void Main(string[] args)

{

Console.Write("Please enter the length of list : ");

int n = Convert.ToInt32(Console.ReadLine());

int[] array = new int[n];

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

{

Console.Write("element [{0}] = ", i);

array[i] = Convert.ToInt32(Console.ReadLine());

}

// Printing of unsorted list

Console.Write("\nUnsorted List is : { ");

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

{

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

}

Console.WriteLine("}");

Console.WriteLine("\nList after applying Bucket Sort algorithm ");

Console.Write("{ ");

BucketSort(ref array);

array.ToList();

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

{

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

}

Console.WriteLine("}");

Console.WriteLine("\nHave a nice Day !!!");

}

// Bucket Sort Method

public static void BucketSort(ref int[] array)

{

int minimum = int.MaxValue;

int maximum = 0;

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

{

if (array[i] < minimum)

minimum = array[i];

if (array[i] > maximum)

maximum = array[i];

}

List<int>[] bucket = new List<int>[maximum - minimum + 1];

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

{

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

}

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

{

bucket[array[i] - minimum].Add(array[i]);

}

int k = 0;

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

{

if (bucket[i].Count > 0)

{

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

{

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

k++;

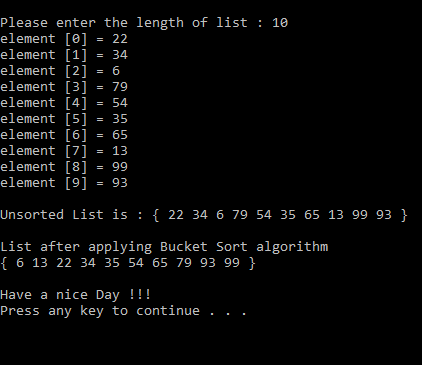
}

}

}

}

**Output:**



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

**Solution:**

**Binary Class:**

class Binary{

public int root = 0;

public string[] str = new string[10];

public void Root(string key){

str[0] = key;}

public void print(){

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

if (str[i] != null){

Console.Write(str[i] + " ");}

else{

Console.Write("-");}}

Console.WriteLine();}

public void Left(string key, int root\_index){

int left\_index = (root\_index \* 2) + 1;

if (str[root\_index] == null){

Console.WriteLine("This " + root\_index + " parent key is not exist");}

else{

str[left\_index] = key;}}

public void Right(string key, int root\_index){

int right\_index = (root\_index \* 2) + 2;

if (str[root\_index] == null){

Console.WriteLine("This " + root\_index + " parent key is not exist");}

else{

str[right\_index] = key; }}

public void InOrder(int root){

if (root < str.Length){

InOrder((root \* 2) + 1);

Console.Write(str[root] + " ");

InOrder((root \* 2) + 2); }}

public void PreOrder(int root){

if (root < str.Length){

if (str[root] != ""){

Console.Write(str[root] + " ");

PreOrder((root \* 2) + 1);

PreOrder((root \* 2) + 2);}}}

public void PostOrder(int root){

if (root < str.Length){

if (str[root] != ""){

PostOrder((root \* 2) + 1);

PostOrder((root \* 2) + 2);

Console.Write(str[root] + " ");}}}

public void Search(){

Console.WriteLine("Enter any element ");

string search = Console.ReadLine();

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

if (string.Equals(search, str[i])){

Console.WriteLine("This " + search + " element is exist");

return;}}

Console.WriteLine("Sorry, this " + search + " element is not exist");}}

**Main Class:**

Binary b = new Binary();

b.Root("s");

b.Left("h",0);

b.Right("o", 0);

b.Left("a", 1);

b.Right("a", 1);

b.Left("i", 2);

b.Right("b", 2);

b.Left("l", 3);

Console.WriteLine("The Tree is:");

b.print();

Console.WriteLine("\n\*\*\*\*\*\*\*\* InOrder of a tree \*\*\*\*\*\*\*\*");

b.InOrder(0);

Console.WriteLine("\n\n\*\*\*\*\*\*\*\* Pre Order of a tree \*\*\*\*\*\*\*\*");

b.PreOrder(0);

Console.WriteLine("\n\n\*\*\*\*\*\*\*\* Post Order of a tree \*\*\*\*\*\*\*\*");

b.PostOrder(0);

Console.WriteLine("\n\n\*\*\*\*\*\*\*\* Searching \*\*\*\*\*\*\*\*");

b.Search();

**Output:**

