Q: - Write a program to implement concept of Binary Search Tree.

Input:-

Console.WriteLine("Binary Search Tree");

Console.WriteLine("Enter number of values you want to add");

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

int[] array = new int[1000];

int i = 0;

while (i < num)

{

Console.WriteLine("Enter value to add");

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

if (array[i] == 0)

{

array[i] = element;

i++;

continue;

}

else if (array[i] >= element)

{

array[(2 \* i) + 1]=element;

i++;

continue;

}

else

{

array[(2 \* i) + 2]=element;

i++;

continue;

}

}

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

{

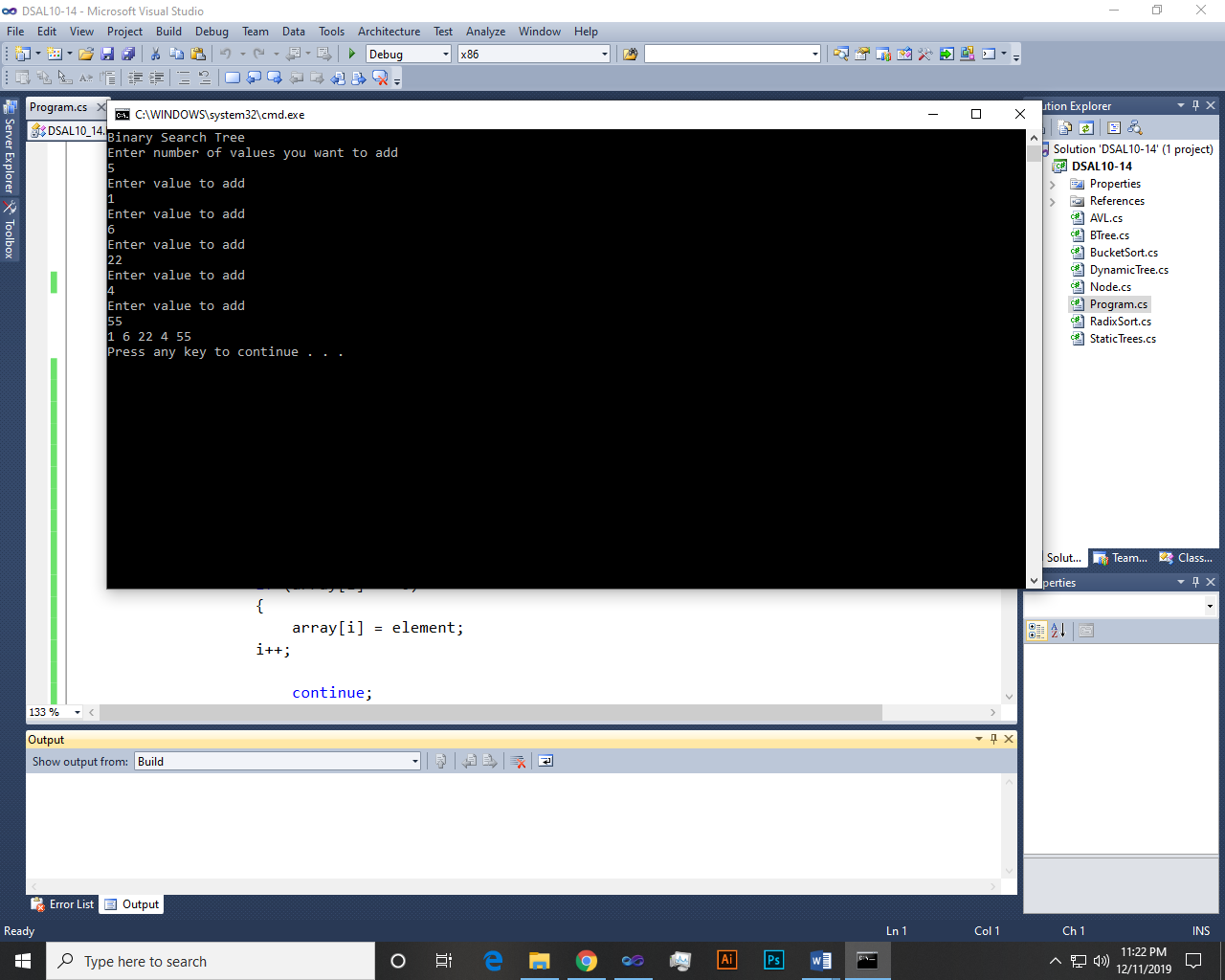
Console.Write(array[j]+" ");

}

Console.WriteLine();

}

Output:-



Q: - Implement the dynamic AVL Tree.

Input:-

AVL:-

Node root;

public AVL()

{

root = new Node();

}

public void Add(int data)

{

Node newItem = new Node(data);

if (root == null)

{

root = newItem;

}

else

{

root = Insert(root, newItem);

}

}

private Node Insert(Node r, Node n)

{

if (r == null)

{

r = n;

return r;

}

else if (n.data < r.data)

{

r.left = Insert(r.left, n);

r = CheckandBalance(r);

}

else if (n.data >= r.data)

{

r.right = Insert(r.right, n);

r = CheckandBalance(r);

}

return r;

}

private Node CheckandBalance(Node temp)

{

int d = difference(temp);

if (d > 1)

{

if (difference(temp.left) > 0)

{

temp = RotateLL(temp);

}

else

{

temp = RotateLR(temp);

}

}

else if (d < -1)

{

if (difference(temp.right) > 0)

{

temp = RotateRL(temp);

}

else

{

temp = RotateRR(temp);

}

}

return temp;

}

public void DeleteRoot(int val)

{

root = Delete(root, val);

}

private Node Delete(Node r, int val)

{

Node parent;

if (r == null)

{ return null; }

else

{

if (val < r.data)

{

r.left = Delete(r.left, val);

if (difference(r) == -2)

{

if (difference(r.right) <= 0)

{

r = RotateRR(r);

}

else

{

r = RotateRL(r);

}

}

}

else if (val > r.data)

{

r.right = Delete(r.right, val);

if (difference(r) == 2)

{

if (difference(r.left) >= 0)

{

r = RotateLL(r);

}

else

{

r = RotateLR(r);

}

}

}

else

{

if (r.right != null)

{

parent = r.right;

while (parent.left != null)

{

parent = parent.left;

}

r.data = parent.data;

r.right = Delete(r.right, parent.data);

if (difference(r) == 2)

{

if (difference(r.left) >= 0)

{

r = RotateLL(r);

}

else { r = RotateLR(r); }

}

}

else

{

return r.left;

}

}

}

return r;

}

public void Find(int key)

{

if (Find(key, root).data == key)

{

Console.WriteLine("{0} was found!", key);

}

else

{

Console.WriteLine("Nothing found!");

}

}

private Node Find(int val, Node current)

{

if (val < current.data)

{

if (val == current.data)

{

return current;

}

else

return Find(val, current.left);

}

else

{

if (val == current.data)

{

return current;

}

else

return Find(val, current.right);

}

}

public void DisplayTree()

{

if (root == null)

{

Console.WriteLine("The tree is empty");

return;

}

InOrder(root);

Console.WriteLine();

}

private void InOrder(Node r)

{

if (r != null)

{

InOrder(r.left);

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

InOrder(r.right);

}

}

private int max(int l, int r)

{

if (l > r)

return l;

else return r;

}

private int depth(Node current)

{

int dep = 0;

if (current != null)

{

int l = depth(current.left);

int r = depth(current.right);

int m = max(l, r);

dep = m + 1;

}

return dep;

}

private int difference(Node current)

{

int l = depth(current.left);

int r = depth(current.right);

int diff = l - r;

return diff;

}

private Node RotateRR(Node parent)

{

Node temp = parent.right;

parent.right = temp.left;

temp.left = parent;

return temp;

}

private Node RotateLL(Node parent)

{

Node temp = parent.left;

parent.left = temp.right;

temp.right = parent;

return temp;

}

private Node RotateLR(Node parent)

{

Node temp = parent.left;

parent.left = RotateRR(temp);

return RotateLL(parent);

}

private Node RotateRL(Node parent)

{

Node temp = parent.right;

parent.right = RotateLL(temp);

return RotateRR(parent);

}

Main:-

Console.WriteLine("AVL tree");

AVL obj = new AVL();

Console.WriteLine("How many values you want to add");

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

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

{

Console.WriteLine("Enter value");

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

obj.Add(temp);

}

Console.WriteLine("Tree");

obj.DisplayTree();

Console.WriteLine("Enter val");

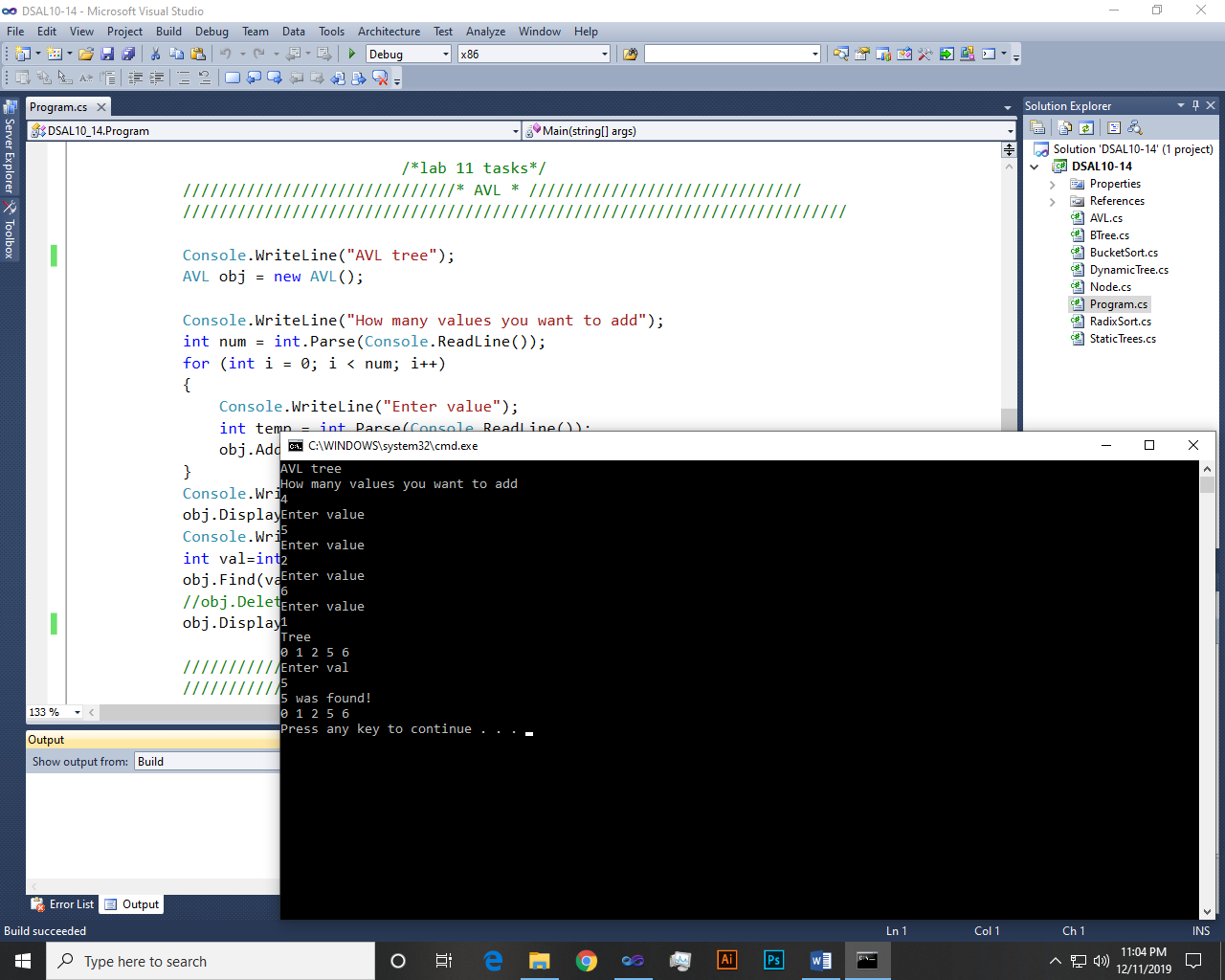
int val=int.Parse(Console.ReadLine());

obj.Find(val);

//obj.DeleteRoot(val);

obj.DisplayTree();

Output:-



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