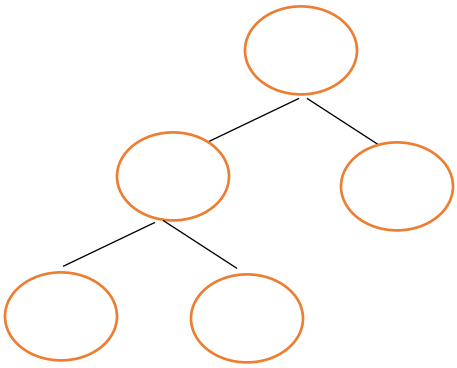
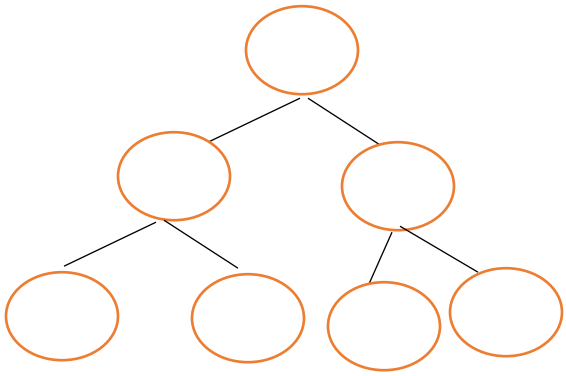
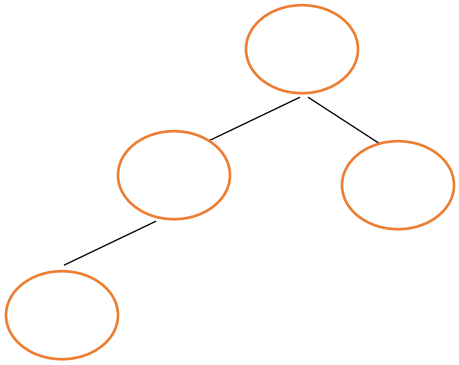
Types of Binary Tree

1- Full binary Tree
If every node has zero or two children.

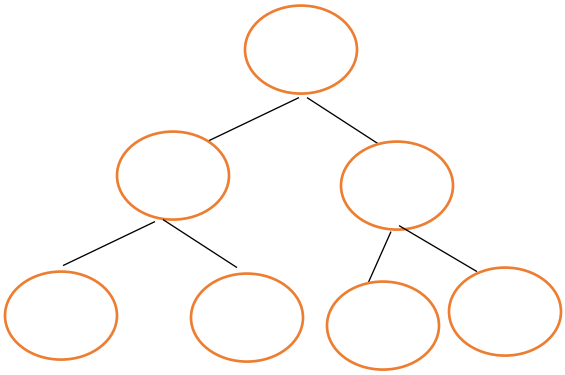


2- Complete binary tree
All levels is completely filled except the last level
All nodes as left as possible in last level



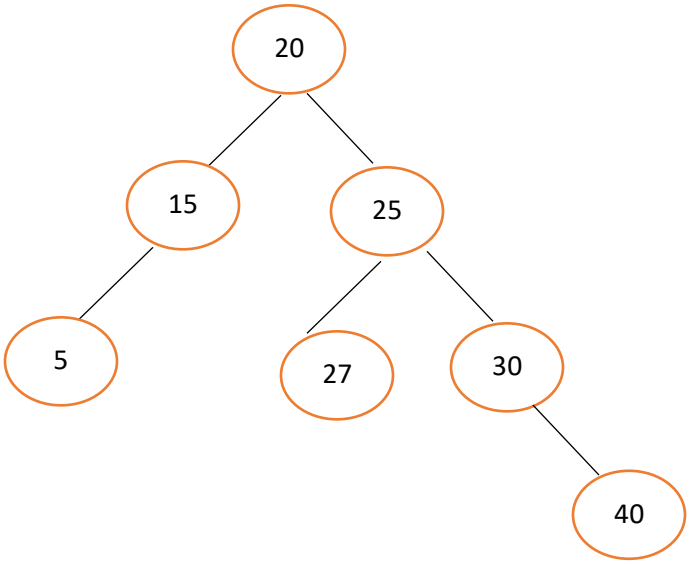


3- Perfect Binary Tree
Every node has two children
All leaves are at the same time



Q) Find the Binary Tree from below numbers, and node 20 is the root.

20, 15 ,5, 25 , 30 , 27 , 40



-maximum number of nodes at level 2^L

- maximum number of nodes in a binary tree $2^{h+1} - 1$

Code of binary tree, insert number, search and traversal.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Reflection.Metadata.Ecma335;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication2
{
   class Program
   {
      class Node
```

```
{
    public int data;
    public Node left, right;

    public Node(int item)
    {
        data = item;
        left = right = null;
    }
}

class BinarySearchTree
{
    Node root;
    public BinarySearchTree()
    {
        root = null;
    }
    public void Insert(int key)
    {
        root =InsertRec(root, key);
    }
    private Node InsertRec(Node root, int key)
    {
        if (root == null)
        {
            root = new Node(key);
            return root;
        }
```

```
        if (key < root.data)
        {
            root.left = InsertRec(root.left, key);
        }
        else if (key > root.data)
        {
            root.right = InsertRec(root.right, key);
        }
        return root;
    }
    public Node Search(int key)
    {
        return SearchRec(root, key);
    }
    private Node SearchRec(Node root, int key)
    {
        if (root == null || root.data == key)
        {
            return root;
        }

        if (key < root.data)
        {
            return SearchRec(root.left, key);
        }
        return SearchRec(root.right, key);
    }
    public void InOrderTraversal()
    {
        InOrderTraversalRec(root);
```

```
    }
    private void InOrderTraversalRec(Node root)
    {
       if (root != null)
       {
          InOrderTraversalRec(root.left);
          Console.Write(root.data + " ");
          InOrderTraversalRec(root.right);
       }
    }
    static void Main(string[] args)
    {
       BinarySearchTree tree = new BinarySearchTree();
       tree.Insert(50);
       tree.Insert(30);
       tree.Insert(20);
       tree.Insert(40);
       tree.Insert(70);
       tree.Insert(60);
       tree.Insert(80);
       Console.WriteLine("Inorder traversal of the binary search tree is:");
      tree.InOrderTraversal();
       int key = 100;
       Node searchResult = tree.Search(key);
       if (searchResult == null)
       {
          Console.WriteLine("\n{key} is not present in the binary search tree.");
       }
       else
       {
```

```
                Console.WriteLine("\n{key} is present in the binary search tree.");
            }
            Console.ReadKey();
        }
    }
  }
}
```