Recursion:

Factorial

```
const factorial = (n) => {
      if(n==1)
            return 1;
      return n * factorial (n-1);
}
```

Power of number
number = 8, power 7
pow(8,7) = 8* pow(8,6) =

```
const power = (number, x ) => {

    if(x == 1)
        return number;
    else
        return number * power(number, x-1);
}

console.log(power(8,3));
```

find if the postive number is in power of 2 using recursion
Ex- 16 = pow(2,4) -> true
24 = -> false

```
const isInPowOfTwo = (number) => {

    if(number == 1)
      return  true;
    return recursive(number, 2);
}

const recursive = (number, x) => {
    if(x == number)
        return true;
    else if(x > number)
        return false;
    else
        return recursive(number, x* 2);
}



const f2 = (number, x) => {
    if(x == number || number == 1)
        return true;
    else if(x > number)
        return false;
    else
        return f2(number, x* 2);
}


n = 512
console.log(f2(43, 2));
console.log(isInPowOfTwo(512));



//Binary tree
class BTNode {
        constructor(value){
            this.value = value;
            this.left = null;
            this.right = null;
        }
}

rootNode = new BTNode(1);
rootNode.left = new BTNode(2);
rootNode.right =  new BTNode(3);
rootNode.left.left =  new BTNode(4);
rootNode.left.right =  new BTNode(5);


//Do pre-order traversal of Given binary tree using recursion

const preOrderTraversal = (node) => {

    if(node == null)
        return;
    console.log(node.value);
    preOrderTraversal(node.left);
    preOrderTraversal(node.right);
}

preOrderTraversal(rootNode);
```