# 2020-04-11 - Handout – Recursion

### Q1. Permutations

Link: https://leetcode.com/problems/permutations/

Given a collection of distinct integers, return all possible permutations.

Example:

```
Input: [1,2,3]
Output:
[ [1,2,3],
  [1,3,2],
  [2,1,3],
  [2,3,1],
  [3,1,2],
  [3,2,1]
]
```

Follow-up: https://leetcode.com/problems/permutations-ii/

### Q2. Merge 2 Sorted Lists

Link: https://leetcode.com/problems/merge-two-sorted-lists/

Merge two sorted linked lists and return it as a new list. The new list should be made by splicing together the nodes of the first two lists.

Example:

```
Input: 1->2->4, 1->3->4
Output: 1->1->2->3->4->4
```

Follow-up: https://leetcode.com/problems/merge-k-sorted-lists/

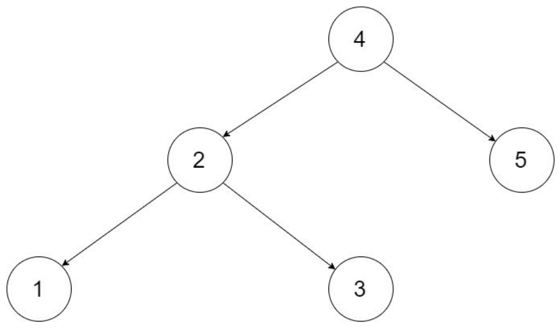
### Q3. Convert Binary Search Tree to Sorted Circular Doubly Linked List in place

Link: https://leetcode.com/problems/convert-binary-search-tree-to-sorted-doubly-linked-list/

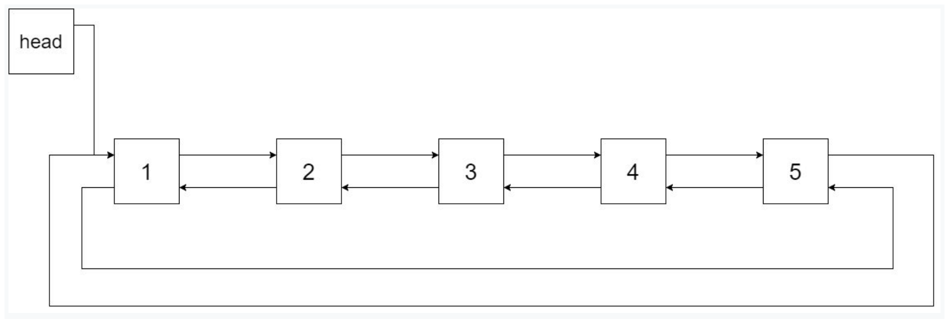
You can think of the left and right pointers as synonymous to the predecessor and successor pointers in a doubly-linked list. For a circular doubly linked list, the predecessor of the first element is the last element, and the successor of the last element is the first element.

We want to do the transformation in place. After the transformation, the left pointer of the tree node should point to its predecessor, and the right pointer should point to its successor. You should return the pointer to the smallest element of the linked list.

Example 1:



`Input: root = [4,2,5,1,3]`



`Output: [1,2,3,4,5]`

Example 2:

Input: root = [2,1,3]

Output: [1,2,3]

Constraints:

- -1000 <= Node.val <= 1000
- Node.left.val < Node.val < Node.right.val
- All values of Node.val are unique.
- 0 <= Number of Nodes <= 2000

**Q4. Remove Invalid Parentheses**

Link: https://leetcode.com/problems/remove-invalid-parentheses/

Remove the minimum number of invalid parentheses in order to make the input string valid. Return all possible results.

Note: The input string may contain letters other than the parentheses ( and ).

Example 1:

Input: "()())()"

Output: ["()()()", "(())()"]

Example 2:

Input: "(a)())()"

Output: ["(a)()()", "(a())()"]

Example 3:

Input: ")("

Output: [""]