Task 1:

You are given an array of integers ‘stones’ where ‘storen[i]’ is the weight of the i-th stone.
We are playing a game with the stones. On each turn, we choose the **heaviest two stones** and smash them together. Suppose the heaviest two stones have weights x and y, with x<=y. The result of the smash is:

- If x==y, both stones are destroyed.
- If x!=y, the stone of weight x is destroyed, and the stone of weight y has a new weight (y-x).

At the end of the game, there is **at most one stone left**. Return the weight of the last remaining stone. If there are no stones left, return 0.

2 7 4 1 8 1 -1	1	Combine 7,8. State: (2 4 1 1 1) Combine 2,4. State: (2 1 1 1) Combine 2,1. State: (1 1 1) Combine 1,1. State: (1) That's the value of the last stone.
10 10 10 10 10 -1	10	
10 10 5 10 10 10 -1	5	
50 30 10 40 20 -1	10	
50 30 10 40 60 20 -1	10	
10 50 30 10 40 60 20 -1	0	
1 7 5 4 2 2 1 4 8 1 -1	1	
1 7 5 4 2 2 1 4 8 -1	0	
3 3 -1	0	
1 -1	1	

Task 2:

Checking parenthesis in Mathematical Expressions

Write a program that will take a mathematical expression as input and check whether it is properly parenthesized or not.

The first line of input will take an integer **N** signifying the number of test cases. The next lines will be **N** mathematical expressions. Each input expression may contain any single-digit number (0~9), operators (+ - x /) and any parenthesis ( )/[ ]/{ }.

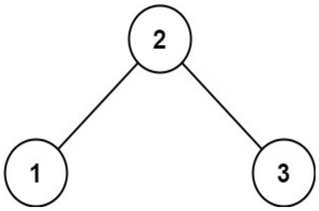
The output will be Yes/No representing whether it is properly parenthesized.

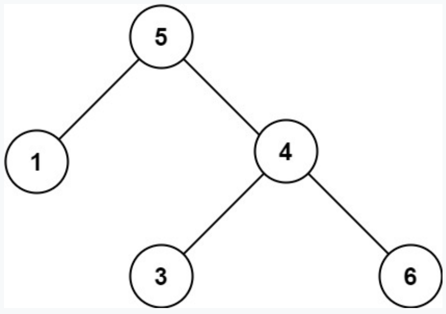
Sample Input	**Sample Output**
8	Yes
[5 + (2 x 5) - (7 / 2)]	No
[1 + { 3 x (2 / 3)] }	Yes
[(1 + 1)]	No
[(1 + 1])	Yes
[( )] { } { [( ) ( )] ( ) }	No
(((	No
[5 + (2 x 5) - (7 / 2)	No
5 + (2 x 5) - (7 / 2)]	No
( )))	No
((( ))	

Task 3:

Given the root of a binary tree, determine if it is a valid binary search tree (BST).A valid BST is defined as follows:

- The left subtree of a node contains only nodes with keys less than the node's key.
- The right subtree of a node contains only nodes with keys greater than the node's key.
- Both the left and right subtrees must also be binary search trees.

Sample Input	**Sample output**	**Explanation**
 root = [2,1,3]	true	

 root =[5,1,4,null,null,3,6]	False	The root node's value is 5 but its right child's value is 4.