Construct Binary Tree from String with bracket representation

Construct a binary tree from a string consisting of parenthesis and integers. The whole input represents a binary tree. It contains an integer followed by zero, one or two pairs of parenthesis. The integer represents the root’s value and a pair of parenthesis contains a child binary tree with the same structure. Always start to construct the left child node of the parent first if it exists.

```
class TreeNode:
    def __init__(self, val):
        self.val = val
        self.left = None
        self.right = None



def buildTree(string):
    if len(string) == 1:
        return TreeNode(int(string[0]))
    return helper(string, 0, len(string) - 1)



def helper(string, si, ei):
    if si >= ei:
        return None
    root = TreeNode(string[si])
    idx = findIndex(string, si + 1, ei)
    root.left = helper(string, si + 2, idx)
    root.right = helper(string, idx + 2, ei)
    return root



def findIndex(string, si, ei):
    if si >= ei:
        return si
    stack = [string[si]]
    si = si + 1
    while len(stack) > 0 and si < ei:
        if string[si] == ')':
            stack.pop()
            if len(stack) == 0:
```

```python
                return si
        elif string[si] == '(':
            stack.append(string[si])
        si += 1
    return si


string = '1(2(3(4(5(6(7(8(9))))))))'
root = buildTree(string)

def preorder(root):
    if root is None:
        return
    print(root.val,end=' ')
    preorder(root.left)
    preorder(root.right)

preorder(root)
```