Stack

```python
stack = list ()

# Append Operation
stack.append ('a')
stack.append ('b')
stack.append ('c')
print ('Initial Stack')
print (stack)
```

```
Initial Stack
['a', 'b', 'c']
```

```python
# Pop Operation
print (stack.pop ())
print (stack.pop ())
print (stack.pop ())
print (stack)
```

```
c
b
a
[]
```

```python
'''
Given a valid parentheses string stringInput, return the nesting depth
of stringInput.
The nesting depth is the maximum number of nested parentheses.

Example 1:
Input: s = "(1+(2*3)+((8)/4))+1"
Output: 3
Explanation:
Digit 8 is inside of 3 nested parentheses in the string.

Example 2:
Input: s = "(1)+((2))+(((3)))"
Output: 3
Explanation:
Digit 3 is inside of 3 nested parentheses in the string.

Example 3:
Input: s = "()(())((()()))"
Output: 3
'''

class StackDepth:
    def maximumDepth (self, stringInput : str) -> int:
        max_depth = 0
        current_depth = 0
        for char in stringInput:
            if char == '(':
                current_depth += 1
                if current_depth > max_depth:
                    max_depth = current_depth
            elif char == ')':
                current_depth -= 1
        return max_depth

# Example
stack_depth = StackDepth()
print(stack_depth.maximumDepth("(1+(2*3)+((8)/4))+1"))
print(stack_depth.maximumDepth("(1)+((2))+(((3)))"))
print(stack_depth.maximumDepth("()(())((()()))"))

3
3
3
```