YASH KASARE 24 STACKS

```
stack = list ()

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

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

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

```
c
b
a
[]
```

```
'''
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
                max_depth = max(max_depth, current_depth)
            elif char == ")":
                current_depth -= 1

        return max_depth


stringInput = input("Enter a valid parentheses string: ")
stack_depth_solver = StackDepth()
print(f"Output: {stack_depth_solver.maximumDepth(stringInput)}")
```

```
Enter a valid parentheses string:  (1+(2*3)+((8)/4))+1
Output: 3
```

Start coding or generate with AI.