772. Basic Calculator III

Implement a basic calculator to evaluate a simple expression string.

The expression string contains only non-negative integers, `+`, `-`, `*`, `/` operators, open `(` and closing parentheses `)` and empty spaces . The integer division should truncate toward zero.

You may assume that the given expression is always valid. All intermediate results will be in the range of `[-2147483648, 2147483647]`.

Some examples:

```
"1 + 1" = 2

" 6-4 / 2 " = 4

"2*(5+5*2)/3+(6/2+8)" = 21

"(2+6* 3+5- (3*14/7+2)*5)+3"=-12
```

Note: Do not use the `eval` built-in library function.

```
class Solution:
    def calculate(self, s):
        precedence = {
            '+': 1,
            '-': 1,
            '*': 3,
            '/': 3
        }
        s = ''.join(s.split(' '))
        number = []
        operators = []
        i = 0
        while i < len(s):
            ch = s[i]
            if ch == '(':
                operators.append(ch)
                i += 1
            elif ch not in precedence and ch!=')':
                num, idx = self.getNumber(i, s)
                number.append(int(num))
```

```
                i = idx
            elif ch == ')':
                while operators[-1] != '(':
                    v2 = number.pop()
                    v1 = number.pop()
                    op = operators.pop()
                    temp = self.evaluate(v1, v2, op)
                    number.append(temp)
                operators.pop()
                i += 1
            elif ch in precedence:
                while len(operators) and operators[-1] != '(' and
precedence[ch] <= precedence[operators[-1]]:
                    v2 = number.pop()
                    v1 = number.pop()
                    op = operators.pop()
                    temp = self.evaluate(v1, v2, op)
                    number.append(temp)
                operators.append(ch)
                i += 1

        while len(operators):
            v2 = number.pop()
            v1 = number.pop()
            op = operators.pop()
            temp = self.evaluate(v1, v2, op)
            number.append(temp)
        return number[-1]

    def getNumber(self, idx, s):
        num = ''
        temp = idx
        while temp < len(s):
            ch = s[temp]
            if ch in {'1', '2', '3', '4', '5', '6', '7', '8', '9', '0'}:
                num += s[temp]
                temp += 1
            else:
                break
        return num, temp

    def evaluate(self, v1, v2, char):
        if char == '+':
```

```
            return v1 + v2
        elif char == '-':
            return v1 - v2
        elif char == '*':
            return v1 * v2
        else:
            return v1 // v2
```