**Approach**

Since no parentheses used here,can evaluate directly

Keep 2 rules in mind:

1. Evaluate left to right
2. Follow order of precedence, that is first evaluate (\*/) and then (+-)

**Algorithm:**

If number:

Update currentNum

elif not a number and not a space(must be an operator):

Process num based on the operator

# if operator: push num to stack based on sign

# if prev-sign is + push num

# if prev-sign is - push -num

# if prev-sign is \* pop()\*num store in stack

# if prev-sign is / pop()/num store in stack

Update operator to current Operator

Update num to 0

Do this until all values processed

Now add all items in stack, return answer

**Code:**

class Solution:

def calculate(self, s: str) -> int:

num = 0

stack = []

opr = '+'

s += '+ 0'

for currentChar in s:

# if a number

if currentChar.isdigit():

# update curNum

num = num \* 10 + int(currentChar)

# otherwise a space or operator

elif currentChar != " ":

# if operator: push num to stack based on sign

# if prev-sign is + push num

# if prev-sign is - push -num

# if prev-sign is \* pop()\*num store in stack

# if prev-sign is / pop()/num store in stack

#print(f"operator found = {currentChar}")

if opr == '+':

stack.append(num)

elif opr == '-':

stack.append(-num)

elif opr == '\*':

op1 = stack.pop()

stack.append(op1 \* num)

elif opr == '/':

op1 = stack.pop()

if (op1>0 and num>0) or (op1<0 and num<0):

stack.append(op1 // num)

else:

r = abs(op1)//abs(num)

stack.append(-r)

# update opr to currentChar(holds new operator for next operation)

opr = currentChar

# update num to 0

num = 0

res = 0

while stack:

res += stack.pop()

return res