Basic Calculator II

Implement a basic calculator to evaluate a simple expression string.

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

You may assume that the given expression is always valid.

Some examples:

```
"3+2*2" = 7
" 3/2 " = 1
" 3+5 / 2 " = 5
```

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

Credits:
Special thanks to @ts for adding this problem and creating all test cases.

Solution 1

```
public class Solution {
public int calculate(String s) {
    int len;
    if(s==null || (len = s.length())==0) return 0;
    Stack<Integer> stack = new Stack<Integer>();
    int num = 0;
    char sign = '+';
    for(int i=0;i<len;i++){
        if(Character.isDigit(s.charAt(i))){
            num = num*10+s.charAt(i)-'0';
        }
        if((!Character.isDigit(s.charAt(i)) &&' '!=s.charAt(i)) || i==len-1){
            if(sign=='-'){
                stack.push(-num);
            }
            if(sign=='+'){
                stack.push(num);
            }
            if(sign=='*'){
                stack.push(stack.pop()*num);
            }
            if(sign=='/'){
                stack.push(stack.pop()/num);
            }
            sign = s.charAt(i);
            num = 0;
        }
    }

    int re = 0;
    for(int i:stack){
        re += i;
    }
    return re;
}
```

}

written by abner original link here

Solution 2

If you don't like the `44 - op` ASCII trick, you can use `op == '+' ? 1 : -1` instead. And wow, I didn't know C++ has `or`. I'm a Python guy and wrote that out of habit and only realized it after getting this accepted :-)

```
int calculate(string s) {
    istringstream in('+' + s + '+');
    long long total = 0, term = 0, n;
    char op;
    while (in >> op) {
        if (op == '+' or op == '-') {
            total += term;
            in >> term;
            term *= 44 - op;
        } else {
            in >> n;
            if (op == '*')
                term *= n;
            else
                term /= n;
        }
    }
    return total;
}
```

written by StefanPochmann original link here

Solution 3

```
class Solution {
public:
    int calculate(string s) {
        int result = 0, cur_res = 0;
        char op = '+';
        for(int pos = s.find_first_not_of(' '); pos < s.size(); pos = s.find_firs
t_not_of(' ', pos)) {
            if(isdigit(s[pos])) {
                int tmp = s[pos] - '0';
                while(++pos < s.size() && isdigit(s[pos]))
                    tmp = tmp*10 + (s[pos] - '0');
                switch(op) {
                    case '+' : cur_res += tmp; break;
                    case '-' : cur_res -= tmp; break;
                    case '*' : cur_res *= tmp; break;
                    case '/' : cur_res /= tmp; break;
                }
            }
            else {
                if(s[pos] == '+' || s[pos] == '-') {
                    result += cur_res;
                    cur_res = 0;
                }
                op = s[pos++];
            }
        }
        return result + cur_res;
    }
};
```

written by fastcode original link here