224. Basic Calculator

Add to List
QuestionEditorial Solution
My Submissions

- Total Accepted: **39965**
- Total Submissions: **159463**
- Difficulty: **Hard**
- Contributors: **Admin**

Implement a basic calculator to evaluate a simple expression string.

The expression string may contain open (and closing parentheses), the plus + or minus sign -, **non-negative** integers and empty spaces .

You may assume that the given expression is always valid.

Some examples:

"1 + 1" = 2

" 2-1 + 2 " = 3

"(1+(4+5+2)-3)+(6+8)" = 23

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

Subscribe to see which companies asked this question

227. Basic Calculator II

Add to List
QuestionEditorial Solution
My Submissions

- Total Accepted: **35861**

- Total Submissions: **128983**
- Difficulty: **Medium**
- Contributors: **Admin**

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.

Subscribe to see which companies asked this question

```
//C++
//DemonMikalis
#include <iostream>
#include <stdio.h>
#include <string>
#include <stack>
#include <sstream>
using namespace std;

bool level(char c)
{
	if(c=='(' || c==')') return false;
	else return true;
}

int compute(int a, int b, char op)
{
	if(op == '+')
		return b+a;
```

```
	else if(op == '-')
		return b-a;
}

int calculate(string s) {
	s.erase(remove(s.begin(),s.end(),' '),s.end());
	if(s.length()==0) return 0;
	if(s.length()==1) return s[0] - '0';
	stack<char> ops;
	stack<int> vals;
	for(int i=0;i<s.length();i++)
	{
		if(s[i]>='0' && s[i]<='9')
		{
			string temp = "";
			while(i<s.size() && s[i]>='0' && s[i]<='9')
			{
				temp += s[i]; i++;
			}
			stringstream ss(temp); int temp_val;
			ss>>temp_val;
			vals.push(temp_val);
		}
	    if(s[i] == '(')
		{
			ops.push(s[i]);
		}
	    if (s[i] == ')')
		{
			while(ops.top()!='(')
			{
				int a = vals.top(); vals.pop();
				int b = vals.top(); vals.pop();
				char op = ops.top(); ops.pop();
				int new_val = compute(a,b,op);
				vals.push(new_val);
			}
			ops.pop();
		}
	    if (s[i] == '+' || s[i]== '-')
		{
			while(!ops.empty() && level(ops.top()))
			{
				//(zhewei) i.e., special condition
				// for "(1+(4+5+2)-3)+(6+8)"
				// when tackling "1 + 11 -" compute 1+11 first
				int a = vals.top(); vals.pop();
				int b = vals.top(); vals.pop();
				int new_val = compute(a,b,ops.top());
				vals.push(new_val);
				ops.pop();
			}
			ops.push(s[i]);
		}
	}
	while(!ops.empty() && level(ops.top()))
	{
		int a = vals.top(); vals.pop();
		int b = vals.top(); vals.pop();
		int new_val = compute(a,b,ops.top());
		vals.push(new_val);
		ops.pop();
	}
	return vals.top();
```

```
}

int calculate2(string s) {
	s.erase(remove(s.begin(),s.end(),' '),s.end());
	stack<int> vals;
	char sign = '+';
	int res=0,tmp=0;
	for(int i=0;i<s.length();i++)
	{
		if(isdigit(s[i]))
		{
			tmp = tmp*10 + (s[i]-'0');
		}
		if(!isdigit(s[i]) || i==s.length()-1)
		{
			if(sign == '-') vals.push(-tmp);
			else if(sign == '+') vals.push(tmp);
			else{
				int num;
				if(sign == '*')
				{
					num = vals.top()*tmp;
				}else if (sign == '/')
				{
					num = vals.top()/tmp;
				}
				vals.pop();
				vals.push(num);
			}
			sign = s[i];
			tmp = 0;
		}
	}
	while(!vals.empty())
	{
		res += vals.top();
		vals.pop();
	}
	return res;
}


int main(int argc,char *argv[])
{
	string calstr = "(1+(4+5+2)-3)+(6+8)";
	string calstr2 = "3+5 / 2";
	string calstr3 = "3+2*2";
	int ans = calculate(calstr);
	int ans2 = calculate2(calstr3);
	cout << ans << "---"<<  ans2 << endl;
	return 0;
}
```