224. Basic Calculator

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 .

Example 1:

```
Input: "1 + 1"
Output: 2
```

Example 2:

```
Input: " 2-1 + 2 "
Output: 3
```

Example 3:

```
Input: "(1+(4+5+2)-3)+(6+8)"
Output: 23
```

Note:

- You may assume that the given expression is always valid.
- Do not use the `eval` built-in library function

We will solve this question in a single pass
given a string : `(1+(4+5+2)-3)+(6+8)`
maintain :
sum : 0
sign : 1 //since we have only '+' and '-' , we can use 1 for '+' and -1 for '-'
stk :
when we encounter a number add/subtract it to the sum depending upon the sign
when we encounter a (push sum and sign on to the stack and reset sum = 0 and sign = 1(default)
when we encounter a) : 1. pop sign and multiply it with the existing sum
2. pop sum and add/subtract it to the existing sum

finally return the sum

CODE:

```
int calculate(string s) {
    int sign = 1;
    long int sum = 0;
    stack<int> stk;
    for(int i=0;i<s.length();i++)
      {
        if(s[i] >= '0' && s[i] <= '9'){
            long int tempsum = 0;
           while(i<s.length() && s[i] >= '0' && s[i] <= '9')
           {
           tempsum = tempsum*10 + s[i]-'0';
           i++;
           }
           sum += tempsum*sign;
           i--;
        }
        else if(s[i] == '+')    sign = 1;
        else if(s[i] == '-')    sign = -1;
        else if(s[i] == '('){
           stk.push(sum);
           stk.push(sign);
           sum = 0;
```

```
            sign = 1;
        }
        else if(s[i] == ')'){
            sum = stk.top()*sum;  stk.pop();
            sum += stk.top();stk.pop();
        }
    }
    return sum;
}
```