Brief Guide to Completing Assignment 3

1 Simple Arithmetic Expression

The general form of m_exp in stdin:

$$num_1\ \mathbf{h\_op_1}\ num_2\ \mathbf{h\_op_2}\ \cdots\ num_{n-1}\ \mathbf{h\_op_{n-1}}\ num_n\ \mathbf{end\_character}$$

where **h_op** is * or / and **end_character** can be **l_op** (+ or -) or '\n'.

Evaluation of m_exp:

$$(\cdots((num_1\ \mathbf{h\_op_1}\ num_2)\ \mathbf{h\_op_2}\ num_3)\ \cdots\ num_{n-1})\ \mathbf{h\_op_{n-1}}\ num_n)$$

The general form of s_exp in stdin:

$$m\_exp\ \mathbf{l\_op_1}\ m\_exp_2\ \mathbf{l\_op_2}\ \cdots\ m\_exp_{n-1}\ \mathbf{l\_op_{n-1}}\ m\_exp_n\ \mathbf{end\_character}$$

where **l_op** is + - and **end_character** is '\n'.

Evaluation of s_exp:

$$(\cdots((m\_exp_1\ \mathbf{l\_op_1}\ m\_exp_2)\ \mathbf{l\_op_2}\ m\_exp_3)\ \cdots\ m\_exp_{n-1})\ \mathbf{l\_op_{n-1}}\ m\_exp_n)$$

Notice that the structure of s_exp and m_exp is exactly the same. This suggests that the two functions s_exp() and m_exp() could be implemented in a very silmlar way. However for this assignment, you are to implement m_exp() using recursion and s_exp() using loop.

An s_exp example: 34 * 6 - 7/ 3* 4 +5
The first m_exp: 34*6
The second m_exp: 7/3*4
The third m_exp: 5

2 The m_exp() Function

A simple m_exp example in stdin: (□: space, ↩: return, ⇑: indicate location in stdin where next character will be read)

□ □ 3 4 □ * □ □ 3 □ ↩
⇑

How do we implement m_exp() as a recursive function to evaluate an m_exp?

```
float m_exp(float sub_exp, char op) {
1. check if it is the end of m_exp
   how? check op to see if op is '+', '-', or '\n'
2. if yes, return sub_exp (push back op, why?)
3. if not
    get next num of m_exp from stdin with get_num(): save in next_num
    get next op of m_exp from stdin with get_op(): save in next_op
    find 'sub_exp op next_num': save in next_sub_exp
    return m_exp(next_sub_exp, next_op)
           (next_sub_exp has evaluated one more num
                        than sub_exp of the m_exp!)
           (next_op is next operator of the m_exp!)
}
```

3 An Example for m_exp() Function

A simple m_exp example in stdin:

□ □ 3 4 □ * □ □ 3 □ ↩
⇑

We can start by getting first num and first operator
sub_exp = get_num() (sub_exp=34)

□ □ 3 4 □ * □ □ 3 □ ↩
⇑
op = get_op() (op='*')

□ □ 3 4 □ * □ □ 3 □ ↩
⇑

Now call m_exp() function
m_exp(sub_exp, op); (sub_exp is 34, op is '*')
op is not '\n'
next_num=get_num() (next_num is 3)

□ □ 3 4 □ * □ □ 3 □ ↩
⇑

next_op=get_op() (next_op is '\n')

□ □ 3 4 □ * □ □ 3 □ ↩
⇑

next_sub_exp=sub_exp*next_num
return m_exp(next_sub_exp, next_op)
(next_sub_exp is 102, next_op is '\n')
next_op is '\n'
push back next_op

□ □ 3 4 □ * □ □ 3 □ ↩
⇑

return 102

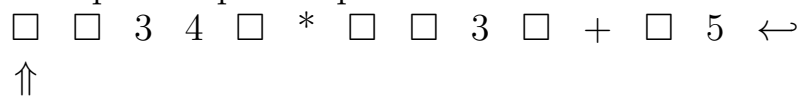
We can also start by calling m_exp() with 1 and '*'

```
m_exp(1,'*');
□ □ 3 4 □ * □ □ 3 □ ↩
⇑
'*' is not '+', '-', or '\'
get_num() ( 34 )

□ □ 3 4 □ * □ □ 3 □ ↩
      ⇑
get_op() ( '*' )

□ □ 3 4 □ * □ □ 3 □ ↩
          ⇑
1*34
return m_exp(34, '*')
        '*' is not '+', '-', or '\'
        get_num() ( 3 )

□ □ 3 4 □ * □ □ 3 □ ↩
              ⇑
        get_op() ( '\n' )

□ □ 3 4 □ * □ □ 3 □ ↩
                    ⇑
        34*3
        return m_exp(102, '\n')
                '\n' is the end
                push back '\n'

□ □ 3 4 □ * □ □ 3 □ ↩
                    ⇑
                return 102
```

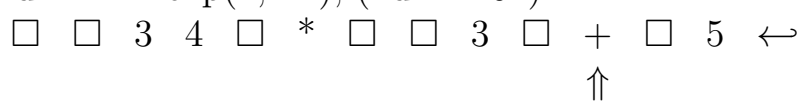
4 The s_exp() Function

How to implement s_exp() with a loop?

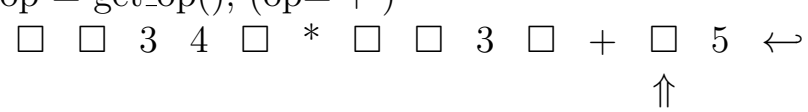
A simple s_exp example in stdin:



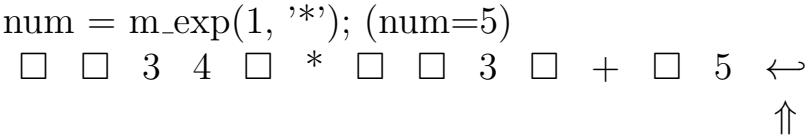
num = m_exp(1, '*'); (num=102)



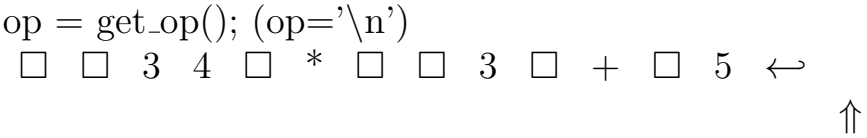
op = get_op(); (op='+')



num = m_exp(1, '*'); (num=5)



op = get_op(); (op='\n')



With m_exp() and get_op(), you can use a while loop or a do while loop to calculate the value of s_exp.

In order for s_exp() to be correct, m_exp() need to push back +, -, or '\n' since those are operators used in s_exp() to perform proper operation or to terminate.

we run s_exp() function as s_exp();

5 Implementation

- Write get_num() and get_op() first and then test these functions before starting the other parts.
- Write m_exp() function and then test it with sample input ending with +, - or '\n' before writing s_exp() function.
- Write s_exp() function and then test it.