Quinn Roemer

CISP - 430

Assignment 5

3/1/2018

Part 0 - Fully Parenthesized Infix Evaluation

Description:

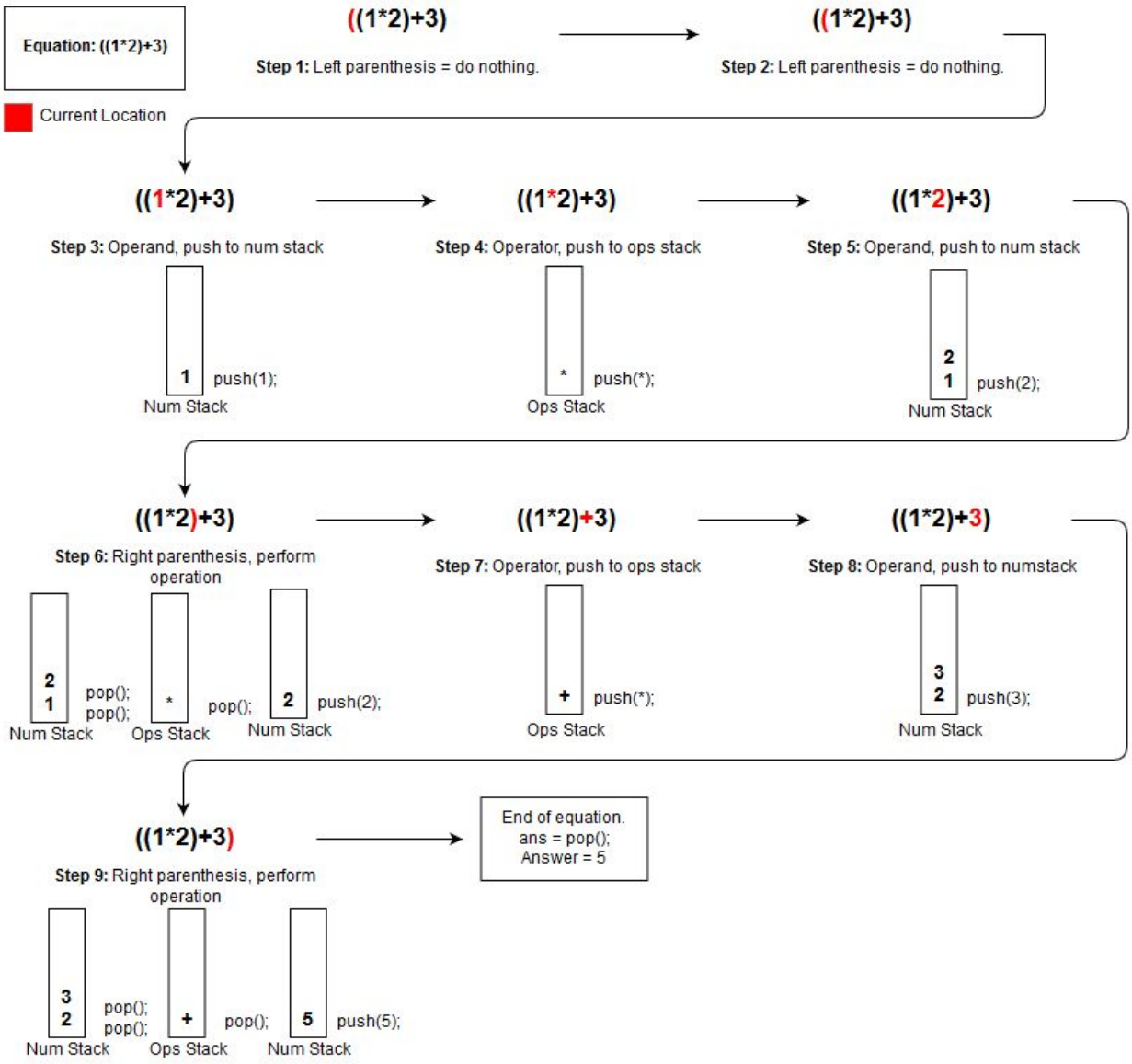
The goal for this part of the assignment was perform a hand execution for the given algorithm for a certain set of mathematical equations. I was to create a detailed diagram that showed how the function worked. This algorithm takes an infix expression and finds the answer.

Function Pseudocode:

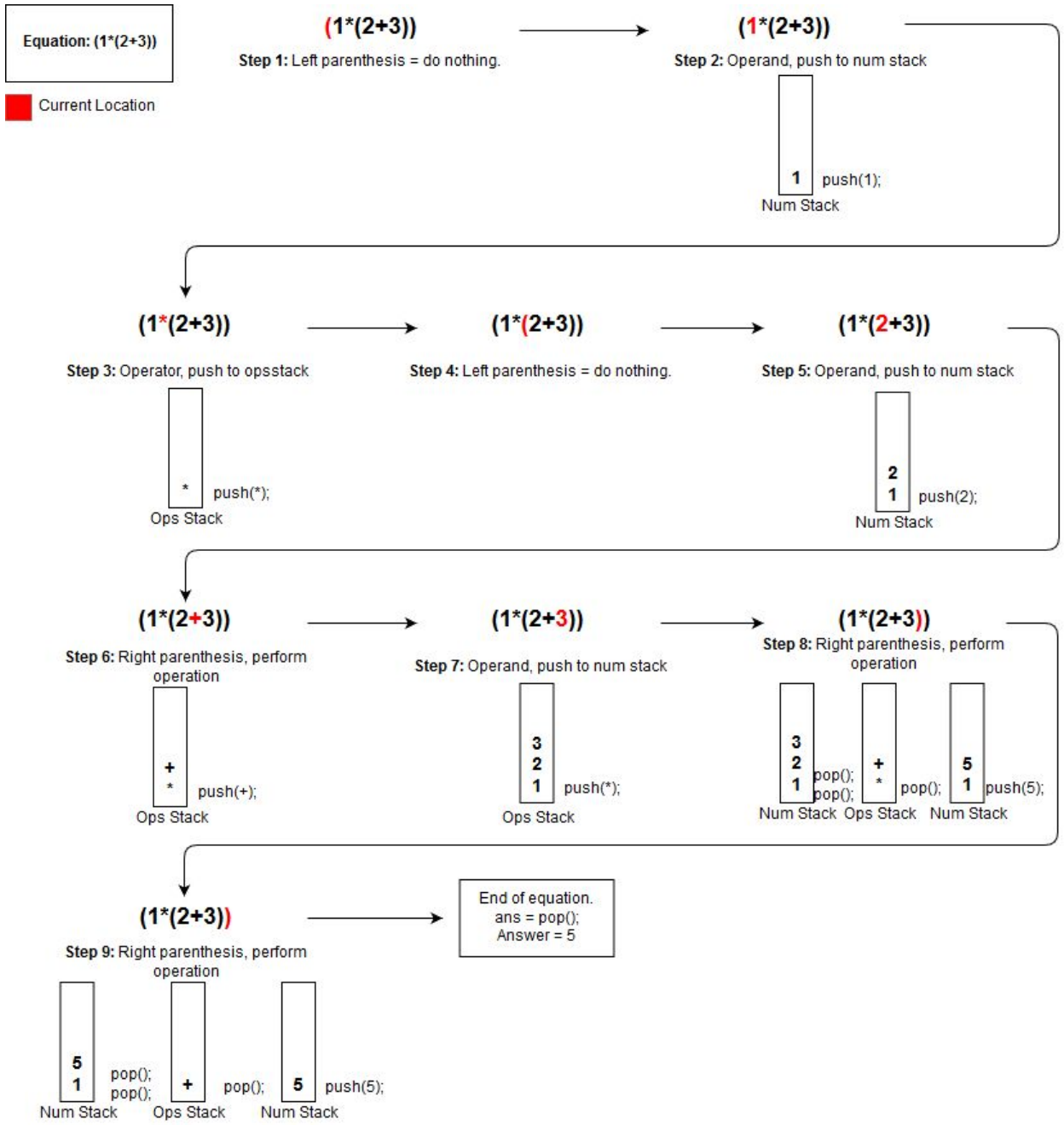
```
while (c=readcharacter)
{
        if (c is a number)
                nums.push(c);
        if (c is an operator)
                ops.push(c);
        if (c is a right parenthesis)
        {
                num1=nums.pop();
                num2=nums.pop();
                op=ops.pop();
                ans=num2 op num1;
                nums.push(ans);
        }
        if (c is a left parenthesis)
        do_nothing;
}
ans = pop();
```

See next page for diagrams.

Hand Execution: ((1*2)+3):



Hand Execution: (1*(2+3)):



Hand Execution: ((2*(3+4))*(5+6)):

The following diagram is very large. As a result, it is placed on the next two pages.

Hand Execution: (((1+2)*(3+4))/(5*(6+(7*(8+9))))):

The following diagram is very large. As a result, it is placed on the next three pages.

Part 1 - Infix to Postfix Conversion

Description:

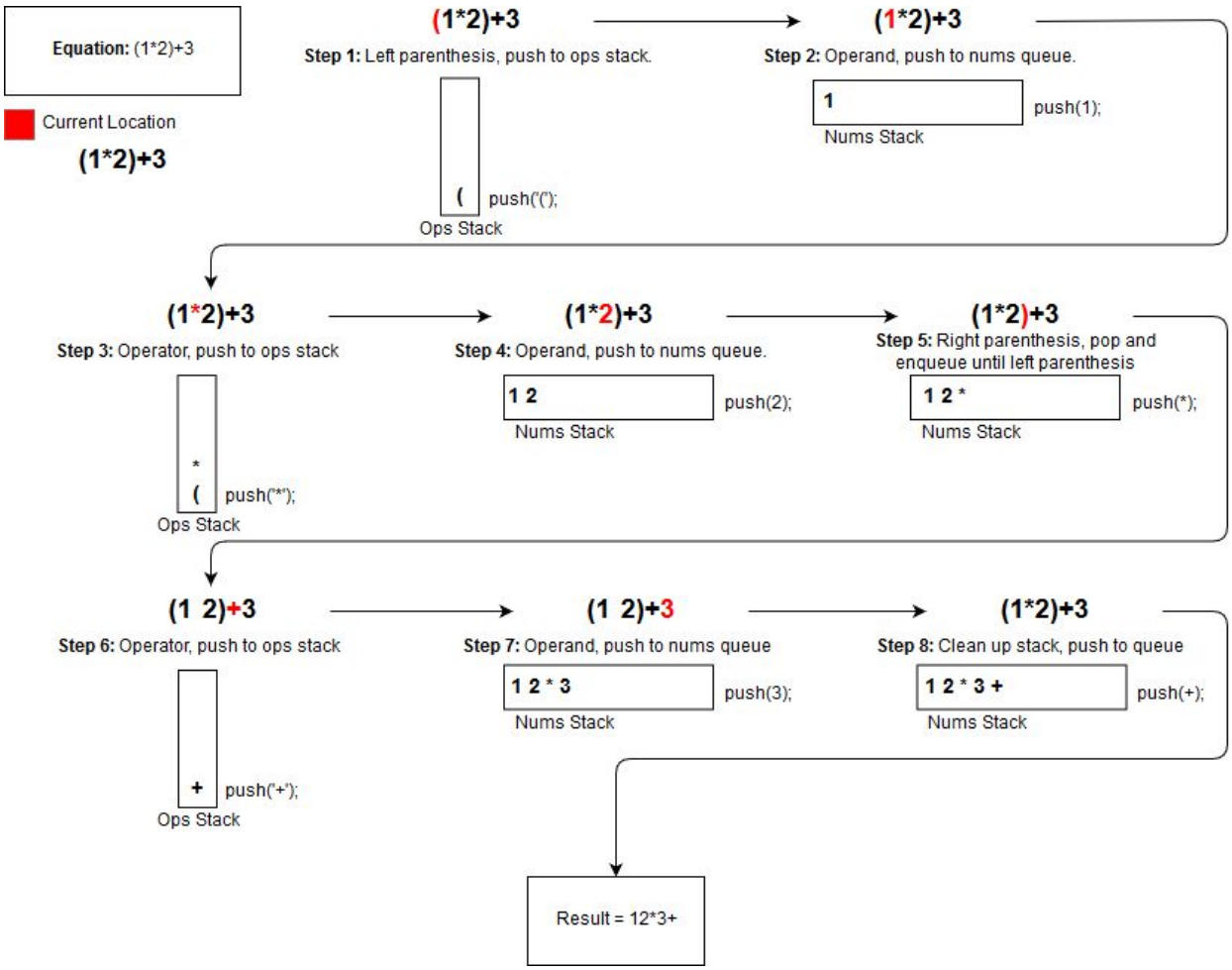
The goal for this part of the assignment was perform a hand execution for the given algorithm for a certain set of mathematical equations. I was to create a detailed diagram that showed how the function worked. This algorithm takes an infix expression and converts it to an equivalent postfix expression.

Function Pseudocode:

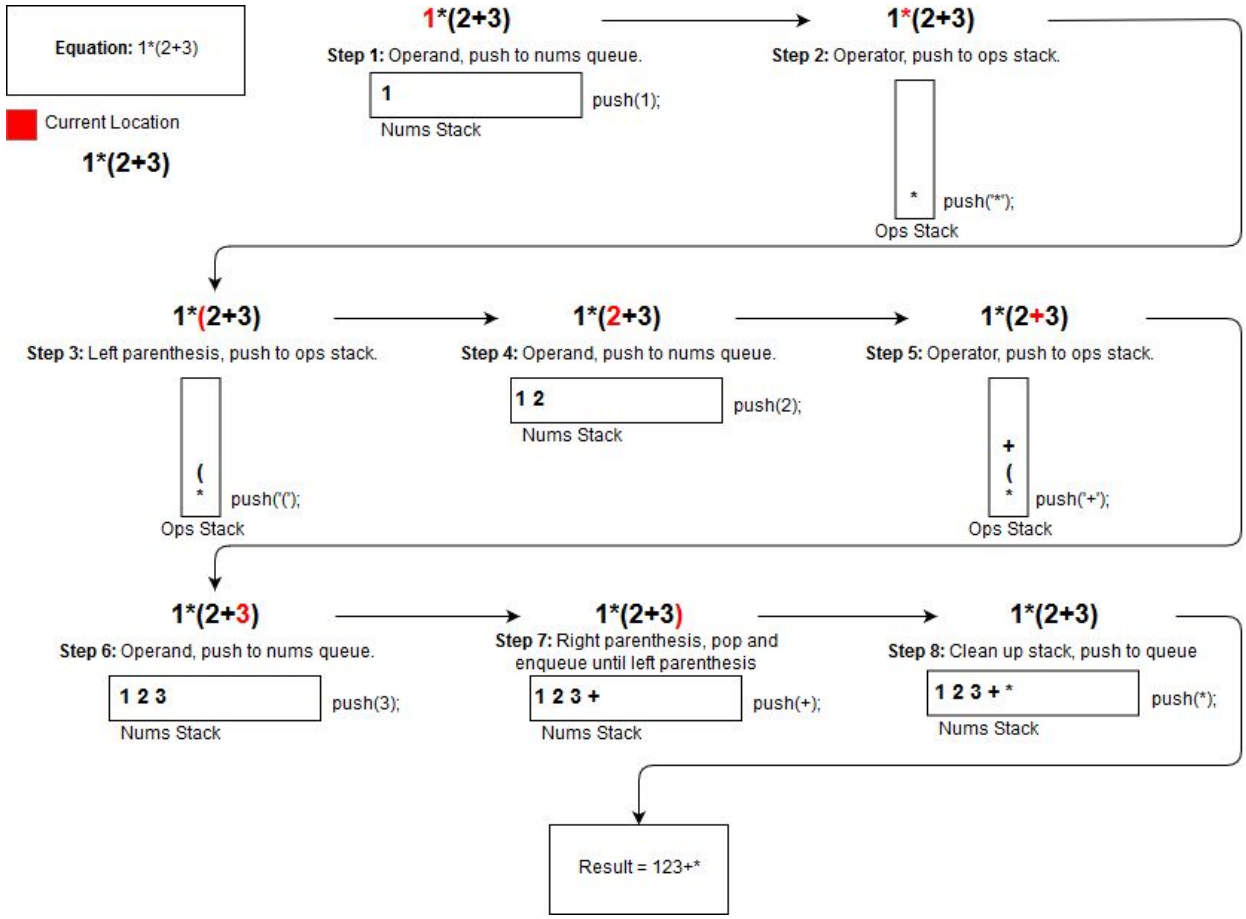
```
//initialize stack and queue to empty;
push('#');
// a dummy operator having a precedence < any other operator (*/, +-, #)
while (not at end of expression)
{
	get a token;
	if (token is an operand)
		enqueue(token);
	else if (token=='(')
		push('(');
	else if (token==')')
	{
		tmp = pop;
		while (tmp != '(')
		{
			enqueue(tmp); tmp = pop;
		}
	}
	else
	{
		while (precedence(token) <= precedence(topofstack))
		{
			tmp = pop; enqueue(tmp);
		}
		push(token);
	}
}
// clean out the stack
while (topofstack != '#')
{
	tmp = pop; enqueue(tmp);
}
```

See next page for diagrams.

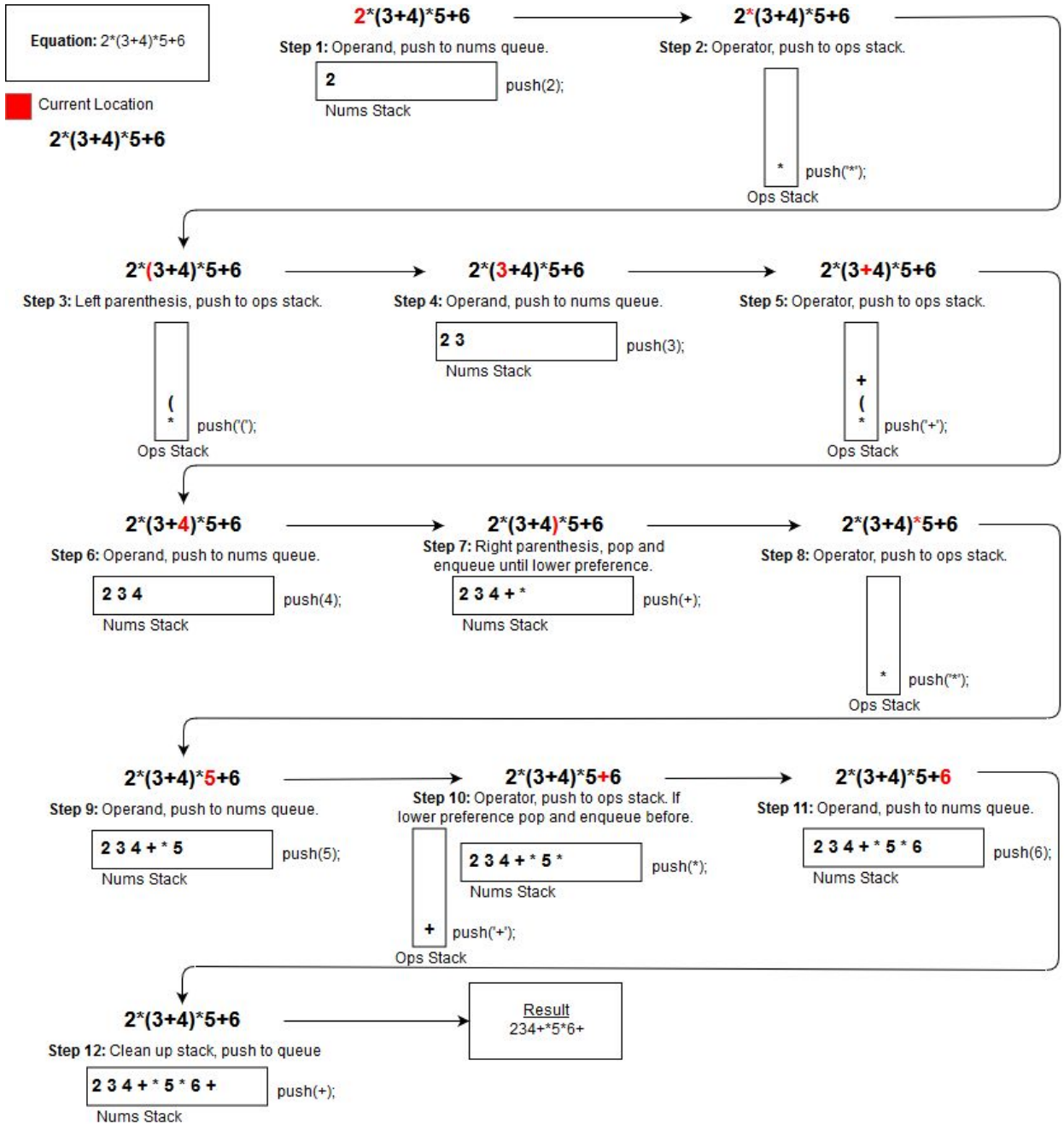
Hand Execution: (1*2)+3:



Hand Execution: 1*(2+3):



Hand Execution: 2*(3+4)*5+6:



Hand Execution: (1+2)*(3+4)/(5*(6+(7*(8+9)))):

The following diagram is very large. As a result, it is placed on the next three pages.

Part 1.1 - Postfix Evaluation

Description:

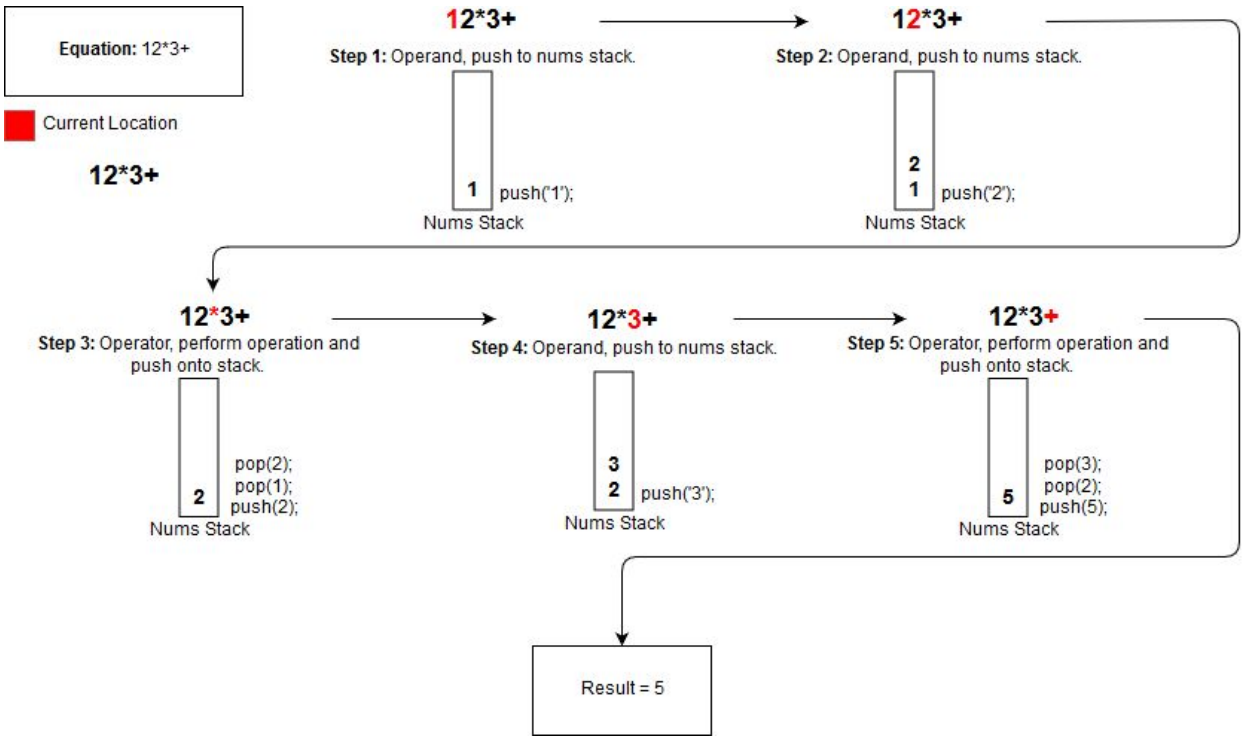
The goal for this part of the assignment was perform a hand execution for the given algorithm for a certain set of postfix mathematical equations. I was to create a detailed diagram that showed how the function worked. This algorithm takes a postfix expression and finds the answer.

Function Pseudocode:

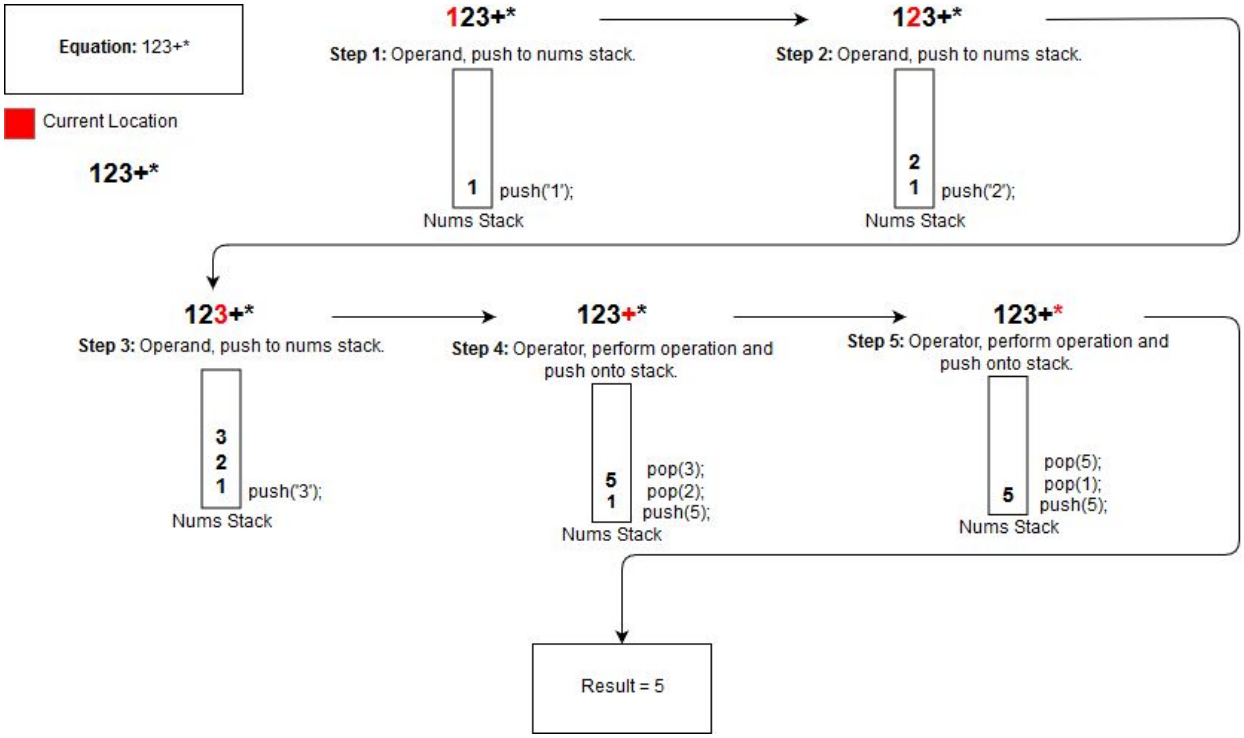
```
//initialize stack to empty;
while (queue not empty)
{
        token = dequeue;
        if (token is an operand)
                push(token)
        else
        {
                operand1=pop; operand2=pop;
                result=perform(operand1,operand2,token);
                push(result);
        }
}
```

See next page for diagrams.

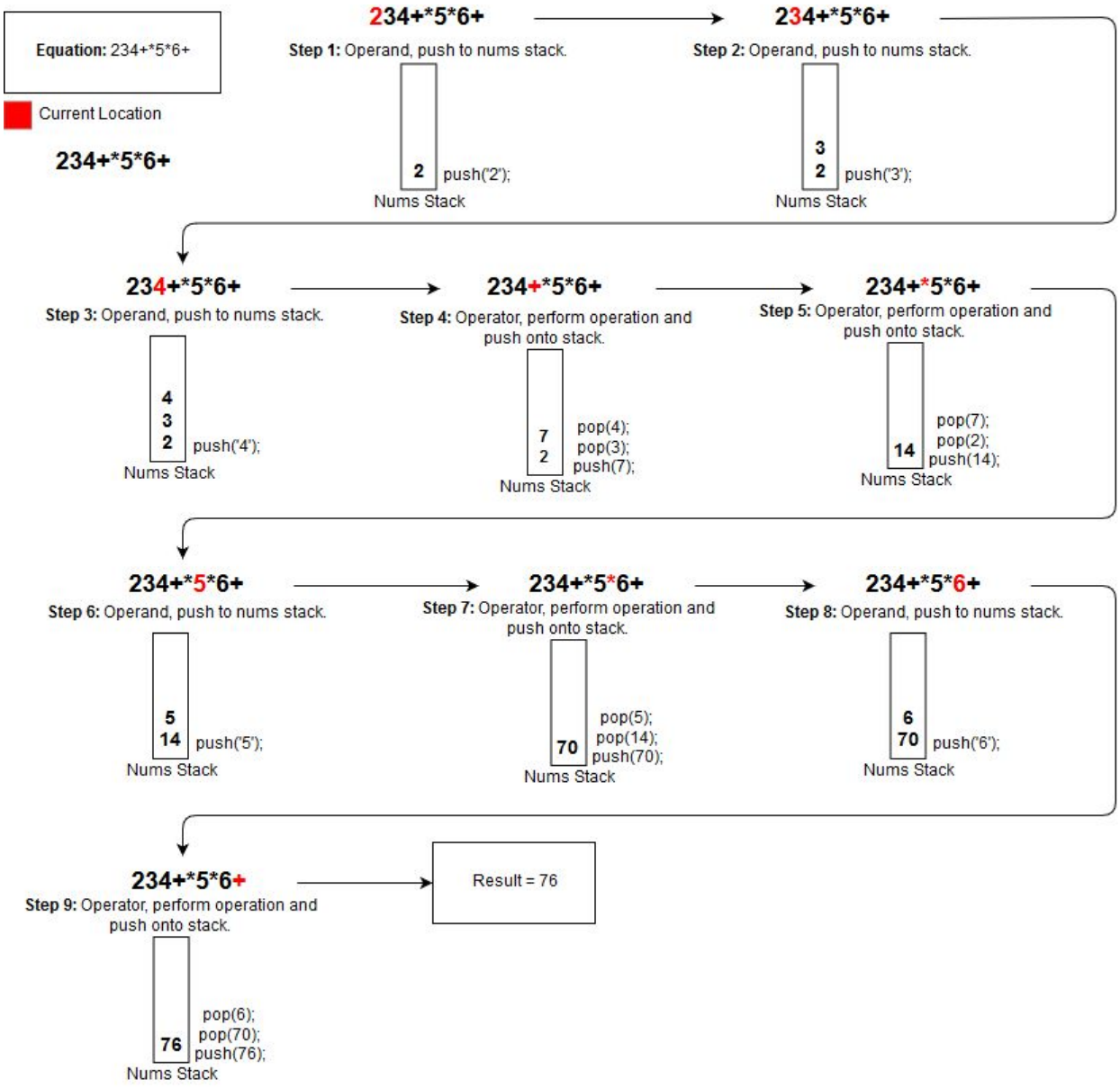
Hand Execution: 12*3+:



Hand Execution: 123+*:



Hand Execution: 234+*5*6+:



Hand Execution: 12+34+*56789+*+*/:

The following diagram is very large. As a result, it is placed on the next two pages.

Conclusion

This was the first time that I have heard about postfix expressions. I never realized that computers evaluate mathematical expressions using this form. I can see how it would be more efficient for a computer to use this form. In infix expressions, a computer would have to look back for the operations while in postfix that is not necessary. This assignment proved to be time-consuming, but was a great learning experience.