	Walker	Posfix stack	Operator Stack	top operator
1	a	a		
2	*	a	*	
3	b	a b	*	
4	/	ab *	/	
5	(	a b *	/ (	
6	c	a b * c	/ (	
7	-	a b *c	/ (-	
8	a	a b *c a	/ (-	
9a	)	a b *c a	/ (	-
9b		a b *c a -	/	(
10	+	a b *c a -/	+	
11	d	a b *c a -/d	+	
12	*	a b *c a -/d	(+) *	
13	e	a b *c a -/d e	(+) *	
14		a b *c a -/d e *	+	
15		a b *c a -/d e * +		

0. Setup walker, post fix stack, operator stack, and top operator.
    a. The walker iterates the given string and evaluates it as a character to perform the next operation.
    b. The post fix stack that is a return value as a post fix notation converted given string.
    c. The operator stack saves a operand and pushes to post fix stack following the notation.
    d. The top operator is a loop condition to decide when the loop stops to push the operator stack proper times.
1. The walker is “a”

    It goes to the postfix stack since it is a variable.
2. The walker is “*”

    It goes to the operator stack since the operator stack is empty.
3. The walker is “b”

    It goes to the postfix stack since it is a variable.
4. The walker is “/”

    It goes to the operator stack and a loop check priority of “*” and “/.” Since the two operands are the same priority, the operator stack pop to the post fix stack and “/” is pushed to the operator stack.
5. The walker is “(“

    It goes to the operator stack since it is “ (“, it goes to operator stack.
6. The walker is “c”

    It goes to the post fix stack since it is a variable.

7. The walker is "-"

   It goes to the operator stack. The loop determines the priority "–" and "(," "-" is a higher priority and it goes to the operator stack without pop.
8. The walker is "a"

   It goes to the post fix stack since it is a variable.
9. The walker is ")"
   a. ")" starts iteration to check the current top operand. It pops the operator stack and stores it as the leading operator. Since it is not "(," the top operator "-" push to post fix stack.
   b. In the second time loop, it pops the operator stack, and it will be "(." It terminates the circle and goes back to the original string iteration.
10. The walker is "+"

    It goes to the operator stack. And it evaluates priority "/" and "+." Since "/" has higher priority, it pops to post fix stack and "+" is pushed to the operator stack.
11. The walker is "d"

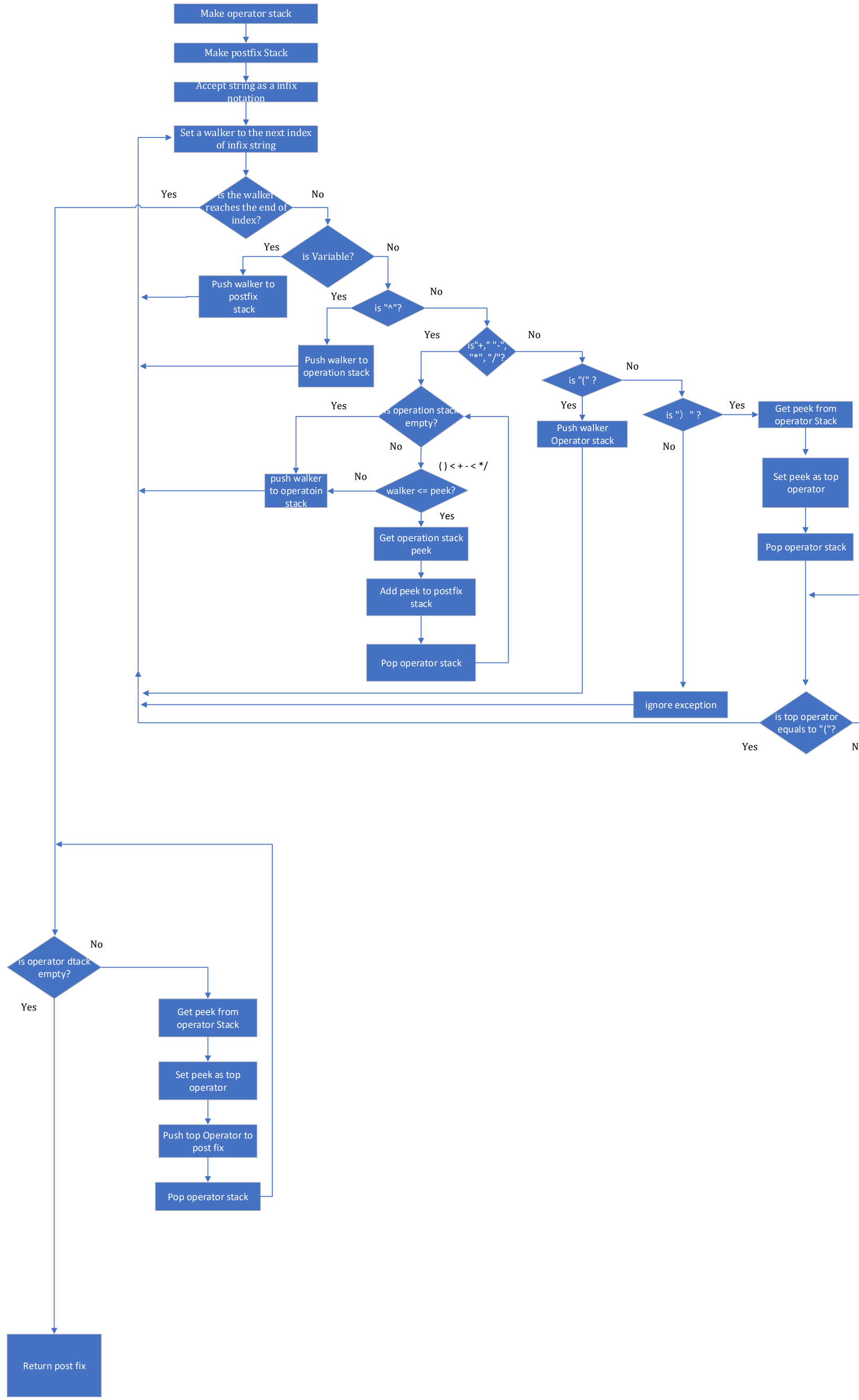
    It goes to the post fix stack since it is a variable.
12. The walker is "*"

    It goes to the operator stack and checks priority "+" and "*." Since "*" is a higher priority, "*" is pushed to the operator stack.
13. The walker is "e"

    It goes to the post fix stack since it is a variable.
14. After given string iteration

    The next time task is that the method iterates the operator stack, pop peeks the operator, and pushes it to the post fix stack. The current peek "*," and it is pushed to the post fix stack.
15. Second iteration of operation stack

    The operation stack still has the "+" operation, which pops and pushes "+" to the post fix operator. And the post fix operator is desired notation.


Make operator stack
Make postfix Stack
Accept string as a infix notation
Set a walker to the next index of infix string
is the walker reaches the end of index?
Yes
No
is Variable?
Yes
No
Push walker to postfix stack
is "^"?
Yes
No
Push walker to operation stack
is"+," "-", "*", "/"?
Yes
No
is operation stack empty?
Yes
No
( ) < + - < */
walker <= peek?
No
Yes
push walker to operatoin stack
Get operation stack peek
Add peek to postfix stack
Pop operator stack
is "(" ?
Yes
No
Push walker Operator stack
is ") " ?
Yes
No
Get peek from operator Stack
Set peek as top operator
Pop operator stack
ignore exception
is top operator equals to "("?
Yes
No
Push top operator to post fix stack
Get peek operator stackas
Set peek as top operator
Pop operator stack
is operator dtack empty?
No
Yes
Get peek from operator Stack
Set peek as top operator
Push top Operator to post fix
Pop operator stack
Return post fix