Evaluation of Prefix Notation and Postfix Notation

Evaluation of Postfix Notation

EVALPOST (P, VALUE)

This algorithm finds the VALUE of an arithmetic expression P written in postfix notation.

```
1. Add a right parenthesis ")" at the end of P. [This acts as a sentinel].
2. Scan P from left to right and Repeat Steps 3 and 4 for each element of P
     until the sentinel ")" is encountered.
3.         If an operand is encountered, put it on STACK.
4.         If an operator * is encountered, then:
           (a) Remove the two top element of STACK, where A is the
                     top element and B is the next-to-top element.
           (b) Evaluate B *A
           (c) Place the result of (b) back on STACK.
   [End of step 2 loop]
5. Set VALUE equal to the top element on STACK.
6. Exit.
```

Evaluation rule of a Postfix Expression states:

While reading the expression from left to right, push the element in the stack if it is an operand.

Pop the two operands from the stack, if the element is an operator and then evaluate it.

Push back the result of the evaluation. Repeat it till the end of the expression.

Algorithm

1) Add) to postfix expression.

2) Read postfix expression Left to Right until) encountered

3) If operand is encountered, push it onto Stack

[End If]

4) If operator is encountered, Pop two elements

i) A -> Top element

ii) B-> Next to Top element

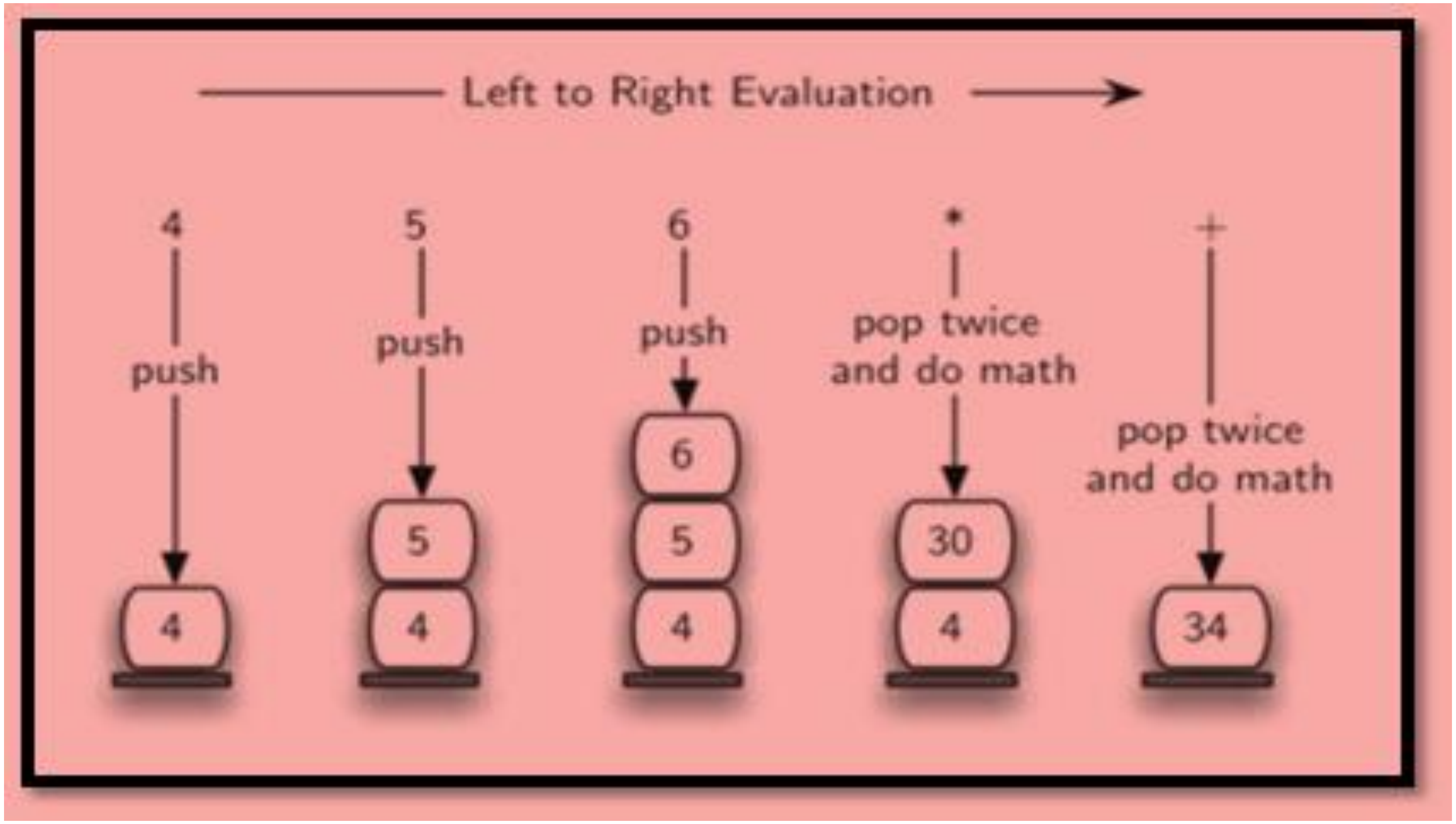
iii) Evaluate B operator A

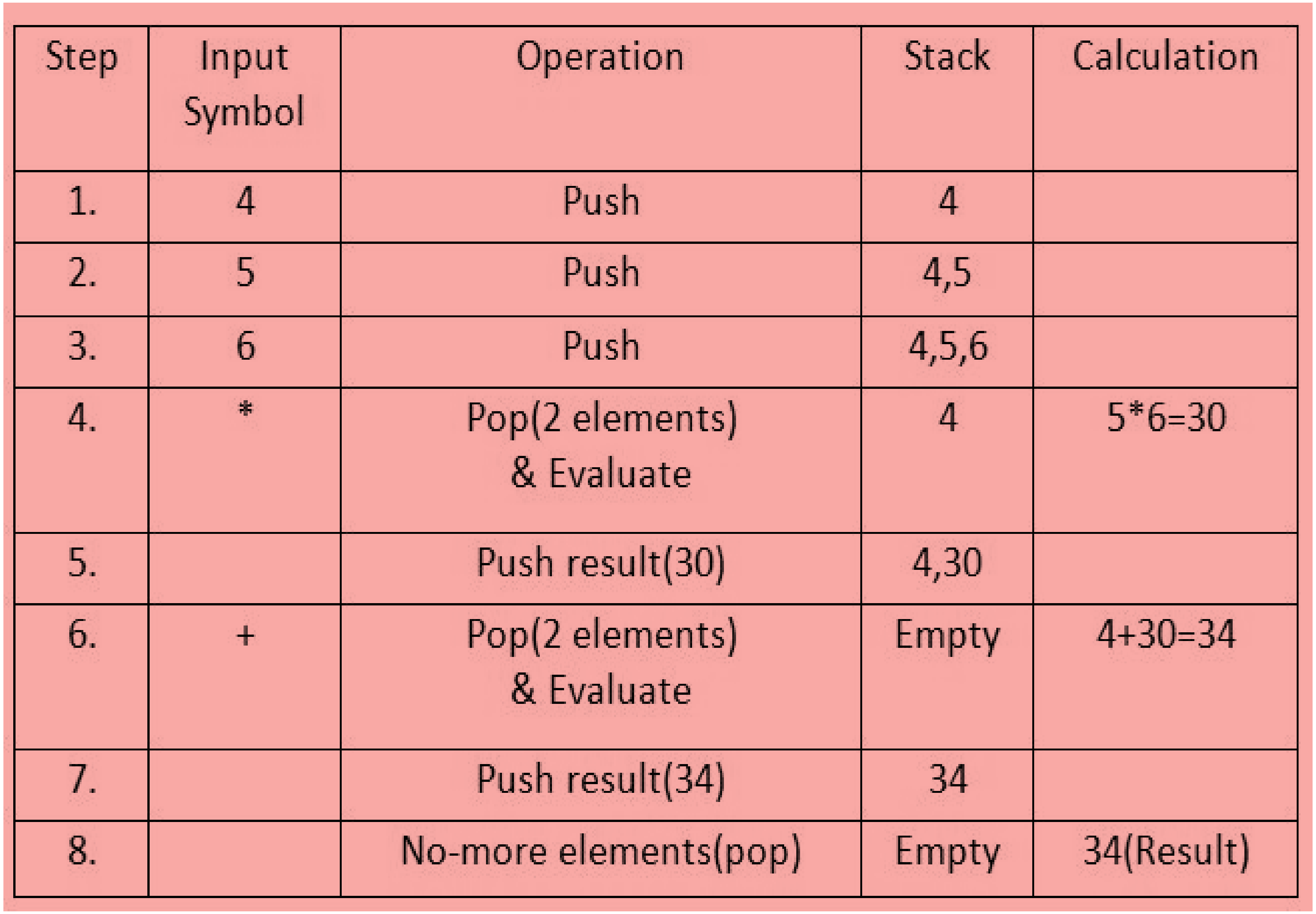
push B operator A onto Stack

5) Set result = pop

6) END

Let's see an example to better understand the algorithm:
Expression: 456*+





Step	Input Symbol	Operation	Stack	Calculation
1.	4	Push	4	
2.	5	Push	4,5	
3.	6	Push	4,5,6	
4.	*	Pop(2 elements) & Evaluate	4	5*6=30
5.		Push result(30)	4,30	
6.	+	Pop(2 elements) & Evaluate	Empty	4+30=34
7.		Push result(34)	34	
8.		No-more elements(pop)	Empty	34(Result)

EVALUATION OF POSTFIX EXPRESSION

- For example, consider the evaluation of the following postfix expression using stacks: **abc+d*f/- where,**
- a=6, b=3, c=6, d=5, f=9 After substituting the values of a, b, c, d and f, the postfix expression becomes 636+5*9/-
- The expression is evaluated as follows:

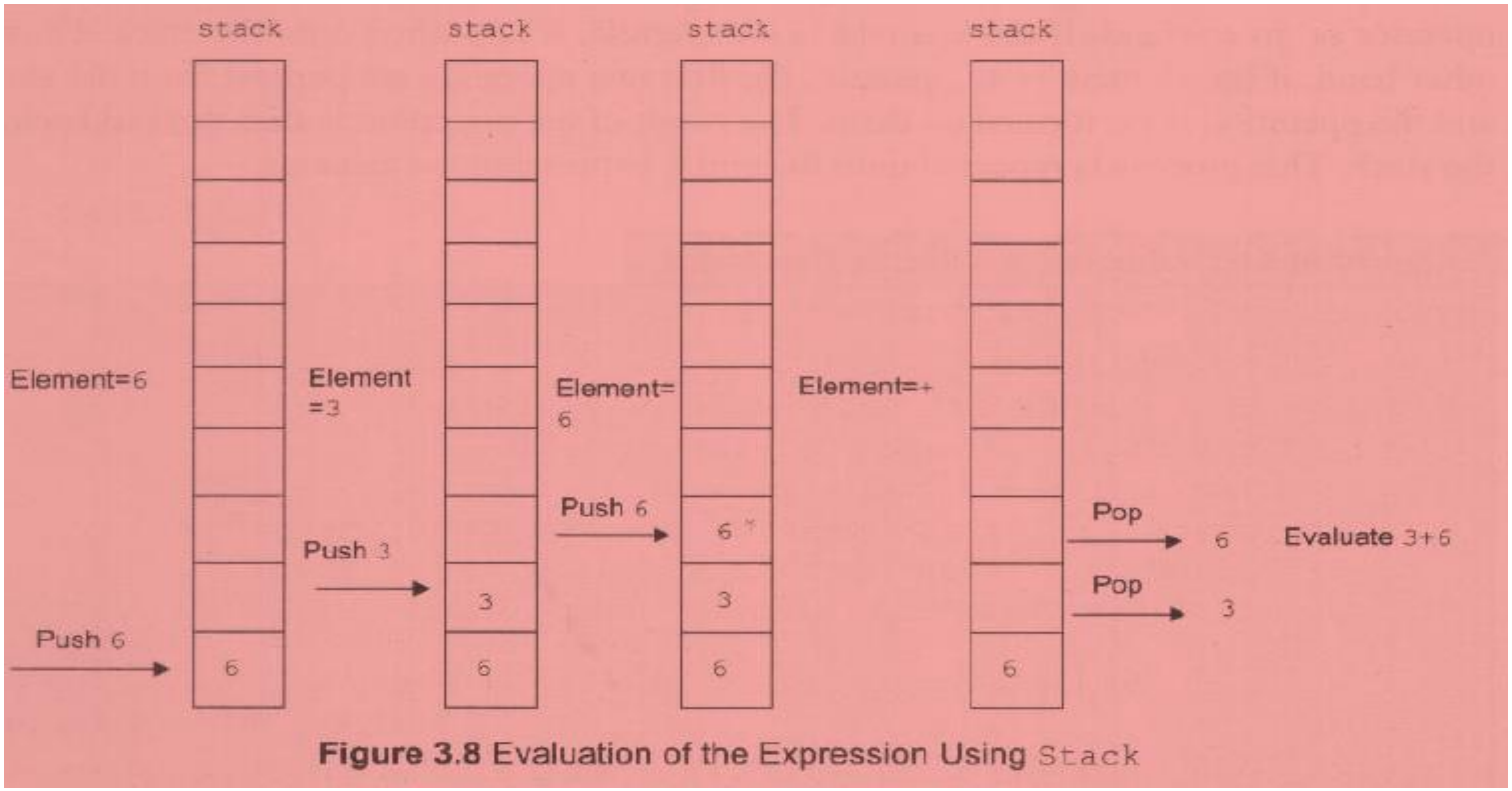


Figure 3.8 Evaluation of the Expression Using Stack

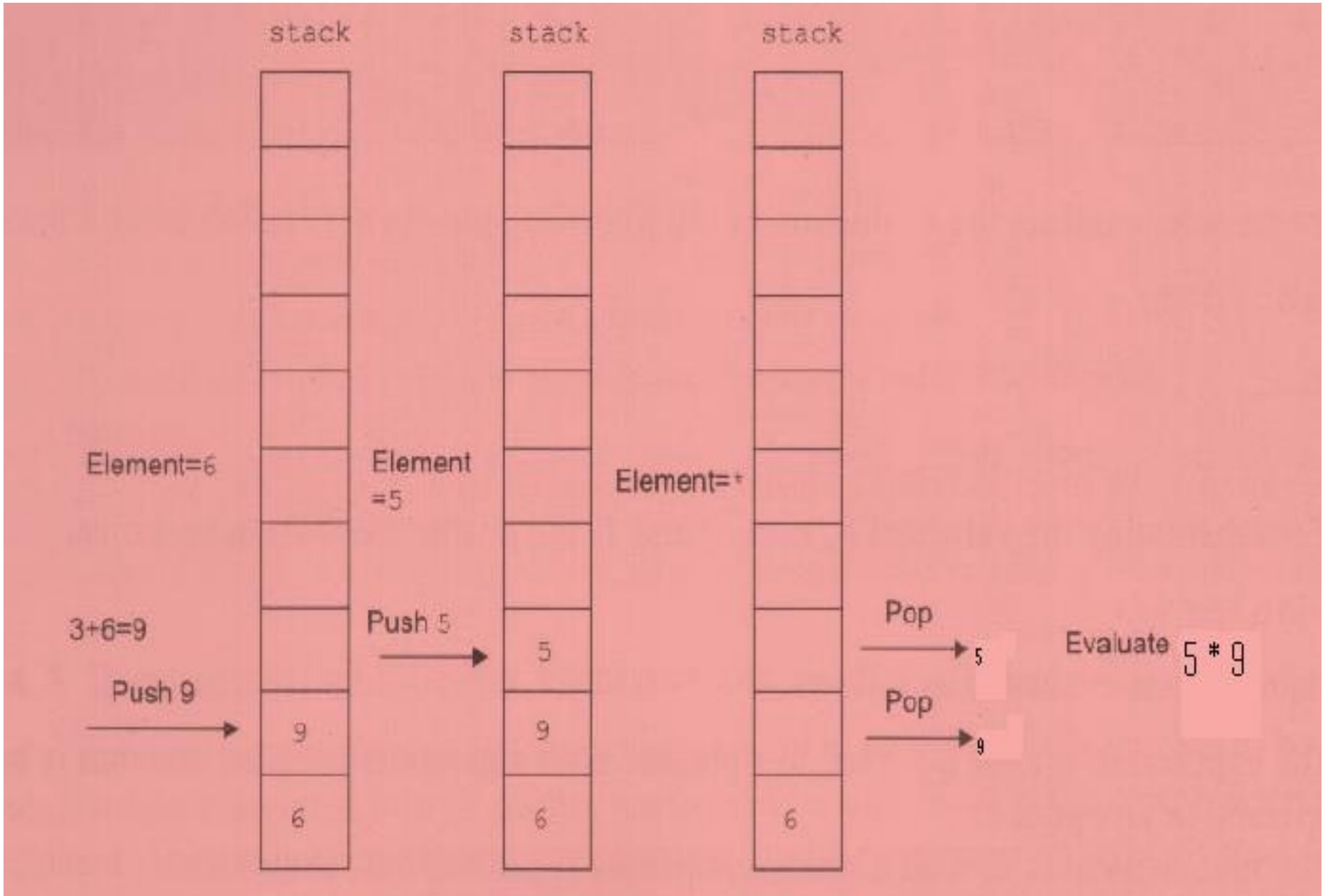
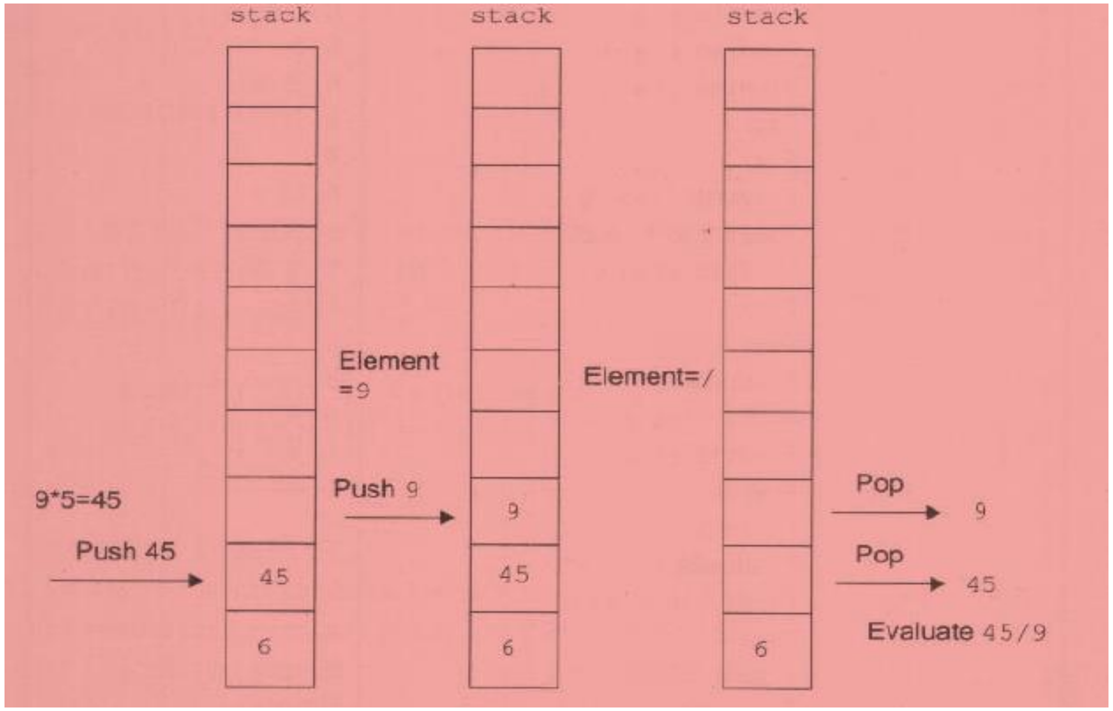


Figure 3.9 Popping 9 and 5 from `Stack`



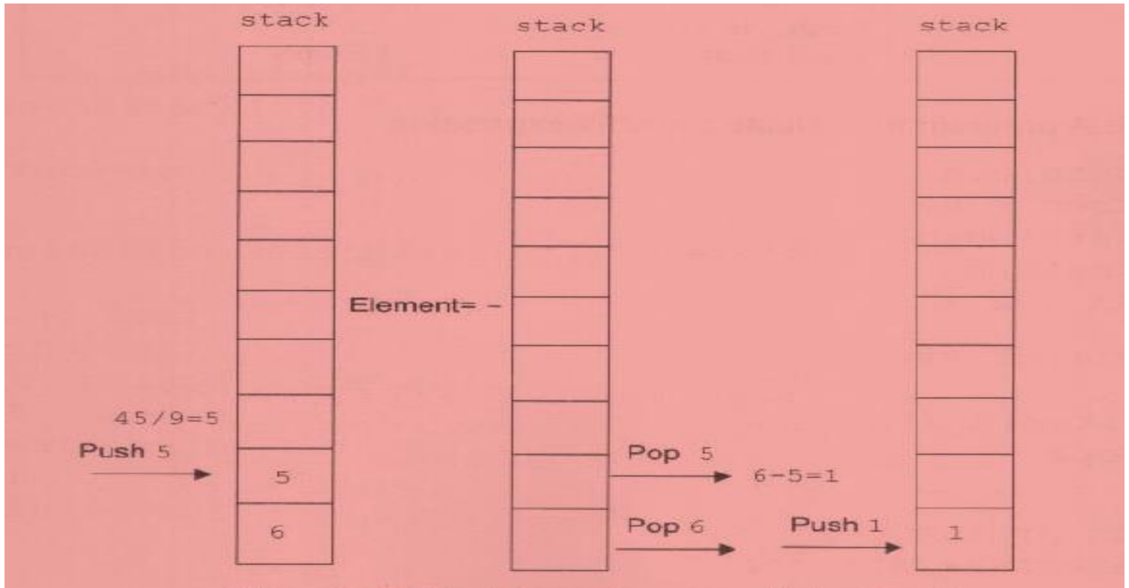
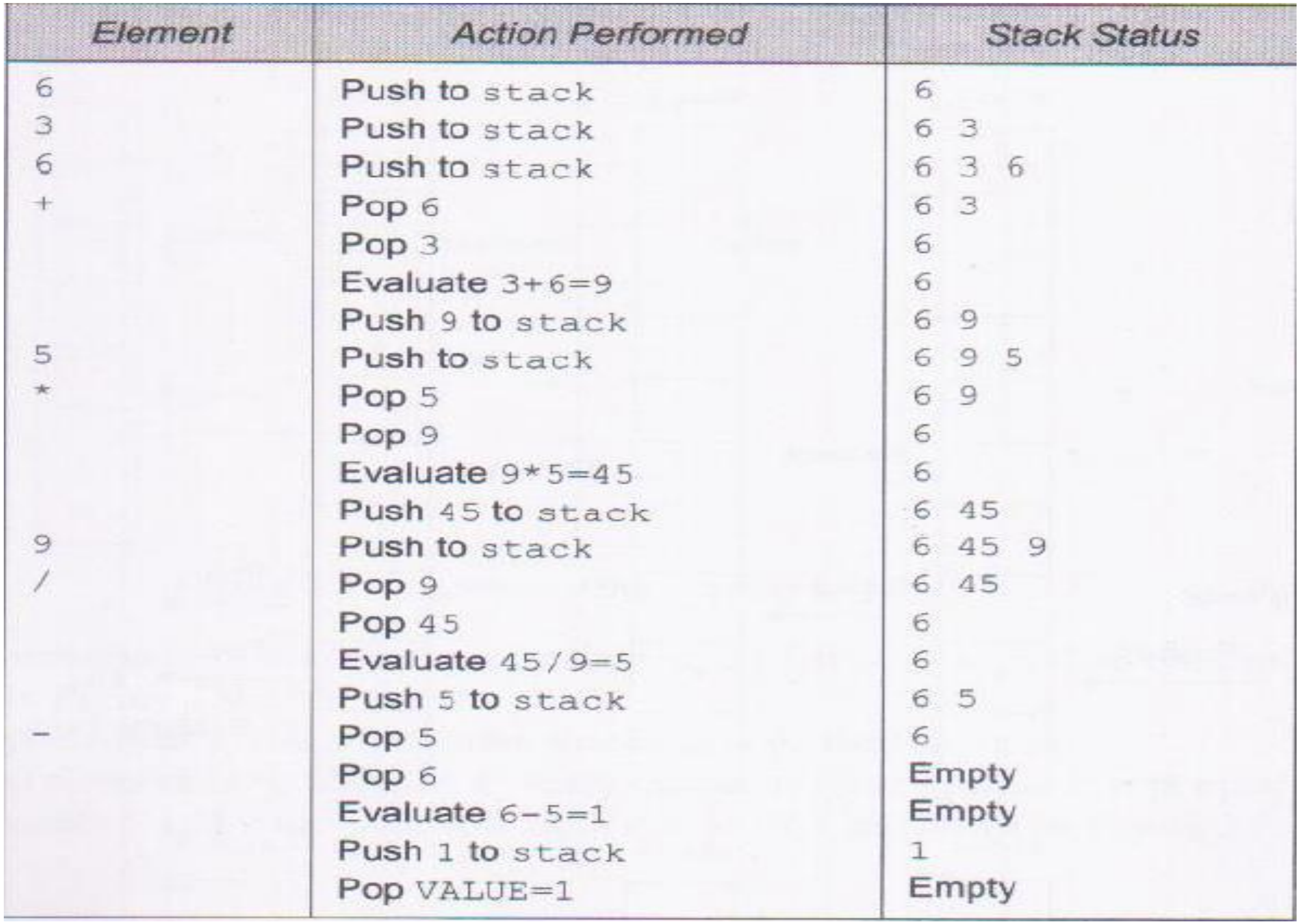


Figure 3.10 Final State of stack with the Result

STEP WISE EVALUATION OF 636+5*9/-



Element	*Action Performed*	*Stack Status*
6	Push to stack	6
3	Push to stack	6 3
6	Push to stack	6 3 6
+	Pop 6	6 3
	Pop 3	6
	Evaluate 3+6=9	6
	Push 9 to stack	6 9
5	Push to stack	6 9 5
*	Pop 5	6 9
	Pop 9	6
	Evaluate 9*5=45	6
	Push 45 to stack	6 45
9	Push to stack	6 45 9
/	Pop 9	6 45
	Pop 45	6
	Evaluate 45/9=5	6
	Push 5 to stack	6 5
–	Pop 5	6
	Pop 6	Empty
	Evaluate 6–5=1	Empty
	Push 1 to stack	1
	Pop VALUE=1	Empty

EXAMPLE:

- **Evaluate Following Postfix Expression:**

(1)A*B+C/D (if A=3,B=2,C=25,D=5) using stack.

(2)AB+C*D(if A=2,B=3,C=4,D=5)(assume + after D)

(3)562+*124/-

(4)ABC^/DE*+AC*-(A=27,B=3,C=2,D=3,E=17)

(5)76 + 4 * 410 - ^ 5 +

(6) 75 – 9 2 / * (ANS: 9.00)

Evaluation of Prefix Notation

```
EVALPRE (P, VALUE)
This algorithm finds the VALUE of an arithmetic expression P written in prefix
notation.
1. Add a left parenthesis "(" at the beg of P. [This acts as a sentinel.]
2. Scan P from right to left and Repeat Steps 3 and 4 for each element of P until the
     sentinel "(" is encountered.
3. IF an operand is encountered, put it on STACK.
4. IF an operator * is encountered, then:
          (a) Remove the two top element of STACK, where A is the Top
                 element and B is the next-to-top element.
          (b) Evaluate A *B.
          (c) Place the result of (b) back on STACK.
   [End of If structure.]
   [End of Step 2 loop.]
5. Set VALUE equal to the top element on STACK.
6. Exit.
```