Stack Applications

Conversion of infix to postfix expression

Figure 5-7 Converting the infix expression a + b * c to postfix form

Next Character in Infix Expression	Postfix Form	Operator Stack (bottom to top)
a	*a*	
+	*a*	+
b	*a b*	+
*	*a b*	+ *
c	*a b c*	+ *
	a b c *	+
	a b c * +	

Figure 5-8 Converting an infix expression to postfix form: a - b + c

Next Character in Infix Expression	Postfix Form	Operator Stack (bottom to top)
a	*a*	
–	*a*	–
b	*a b*	–
+	*a b* –	
	a b –	+
c	*a b* – *c*	+
	a b – *c* +	

Figure 5-8 Converting an infix expression to postfix form: a ^ b ^ c

Next Character in Infix Expression	Postfix Form	Operator Stack (bottom to top)
a	*a*	
^	*a*	^
b	*a b*	^
^	*a b*	^ ^
c	*a b c*	^ ^
	a b c ^	^
	a b c ^ ^	

Infix to Postfix Conversion

1. Operand
   - Append to end of output expression
2. Operator ^
   - Push ^ onto stack
3. Operators +, -, *, /
   - Pop from stack, append to output expression
   - Until stack empty or top operator has lower precedence than new operator
   - Then push new operator onto stack

Infix to Postfix Conversion

4. Open parenthesis
   - Push (onto stack
5. Close parenthesis
   - Pop operators from stack and append to output
   - Until open parenthesis is popped.
   - Discard both parentheses

FIGURE 5-9 The steps in converting the infix expression a / b * (c + (d - e)) to postfix form

Next Character from Infix Expression	Postfix Form	Operator Stack (bottom to top)
a	a	
/	a	/
b	a b	/
*	a b /	
	a b /	*
(	a b /	* (
c	a b / c	* (
+	a b / c	* (+
(	a b / c	* (+ (
d	a b / c d	* (+ (
–	a b / c d	* (+ (–
e	a b / c d e	* (+ (–
)	a b / c d e –	* (+ (
	a b / c d e –	* (+
)	a b / c d e – +	* (
	a b / c d e – +	*
	a b / c d e – + *	

Evaluating Postfix Expressions

- Use a **stack,** assume binary operators +,*
- Input: postfix expression
- Scan the input
  - If operand,
    - **push** to stack
  - If operator
    - **pop** the stack twice
    - apply operator
    - **push** result back to stack

Evaluating Postfix Expressions

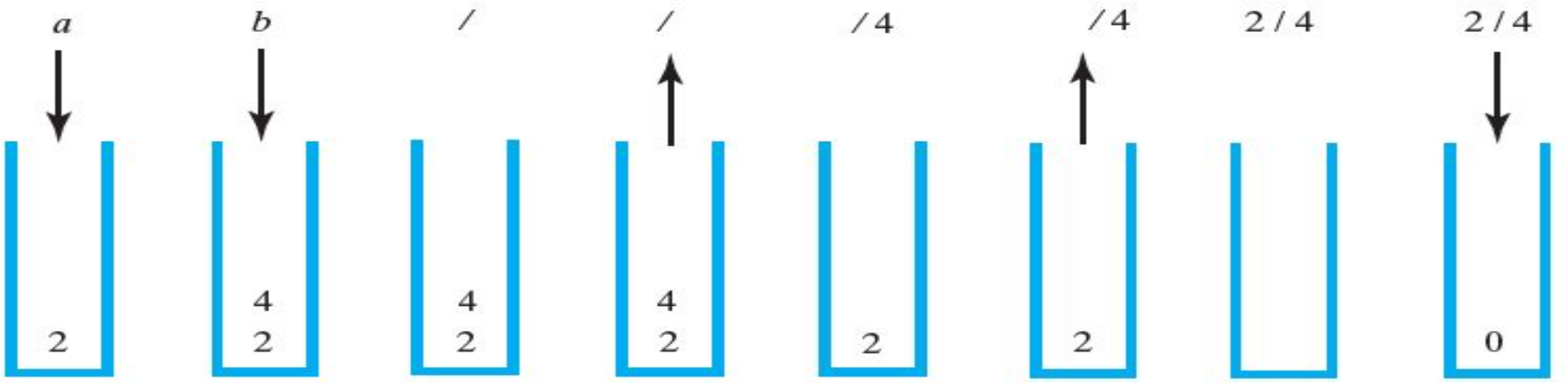

FIGURE 5-10 The stack during the evaluation of the postfix expression a b / when a is 2 and b is 4

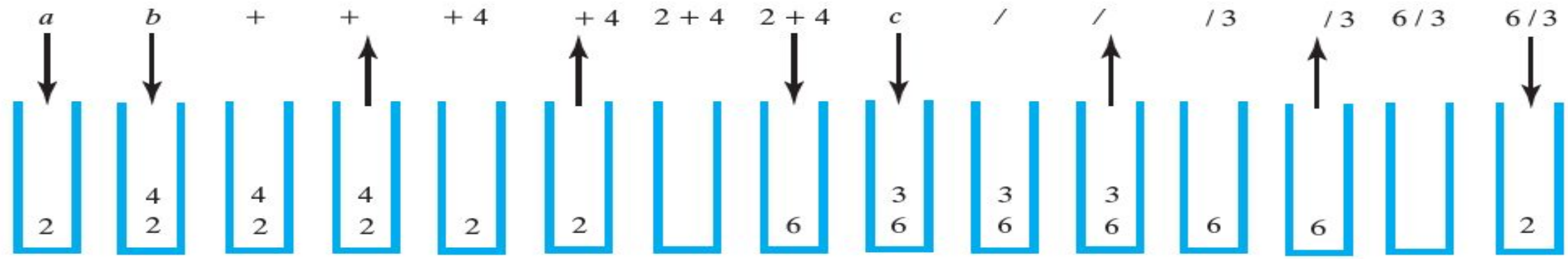

FIGURE 5-11 The stack during the evaluation of the postfix expression a b + c / when a is 2, b is 4, and c is 3

Example

- **Input**
  5 9 8 + 4 6 * * 7 + *
- **Evaluation**
  push(5)
  push(9)
  push(8)
  push(pop() + pop()) /* be careful for '-' */
  push(4)
  push(6)
  push(pop() * pop())
  push(7)
  push(pop() + pop())
  push(pop() * pop())
  print(pop())
- **What is the answer?**

Solve Examples

1)2 3 1*+9-

2)5 3 + 6 2 / * 3 5 * +

3)A B C-D * + E$ F + A=6 B=3 C=2 D=5 E=1 F=7