Stack & queue

Tran Thanh Tung

Kind of structure

Data storage structure that is

- Used in database application

  Array, linked lists, trees, etc

  → Easy to Insert, Delete and Search

- Used as programmer’s tools

  Stacks, queues, etc

  → Restricted access

  → More abstract: underlying mechanism can be array or list, ..

Outline

- Stacks
- Queues
- Priority Queues
- Parsing Arithmetic Expressions

Stack

Introduction

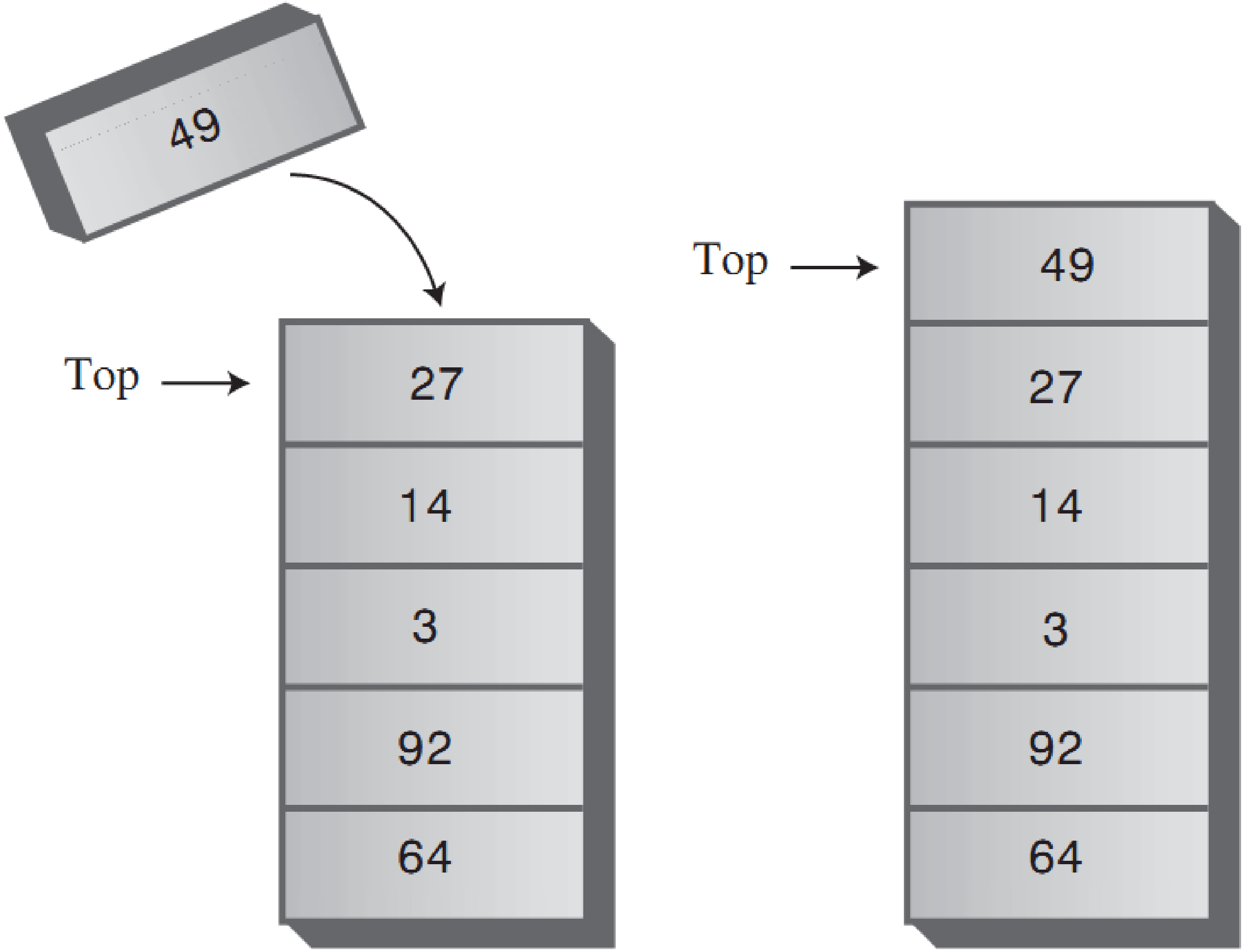
- Accessible item ?
  - →Last inserted item
  - →Last in, first out (LIFO)
- Operations
  - Push ?
  - Pop ?
  - Peek ?
- Properties
  - Stack Overflow (Full)
  - Stack Underflow (Empty)

Stack info

- Info must be managed?
  - Stack size (is full?)
  - Number of element (is empty?)
  - Top item (accessible item)

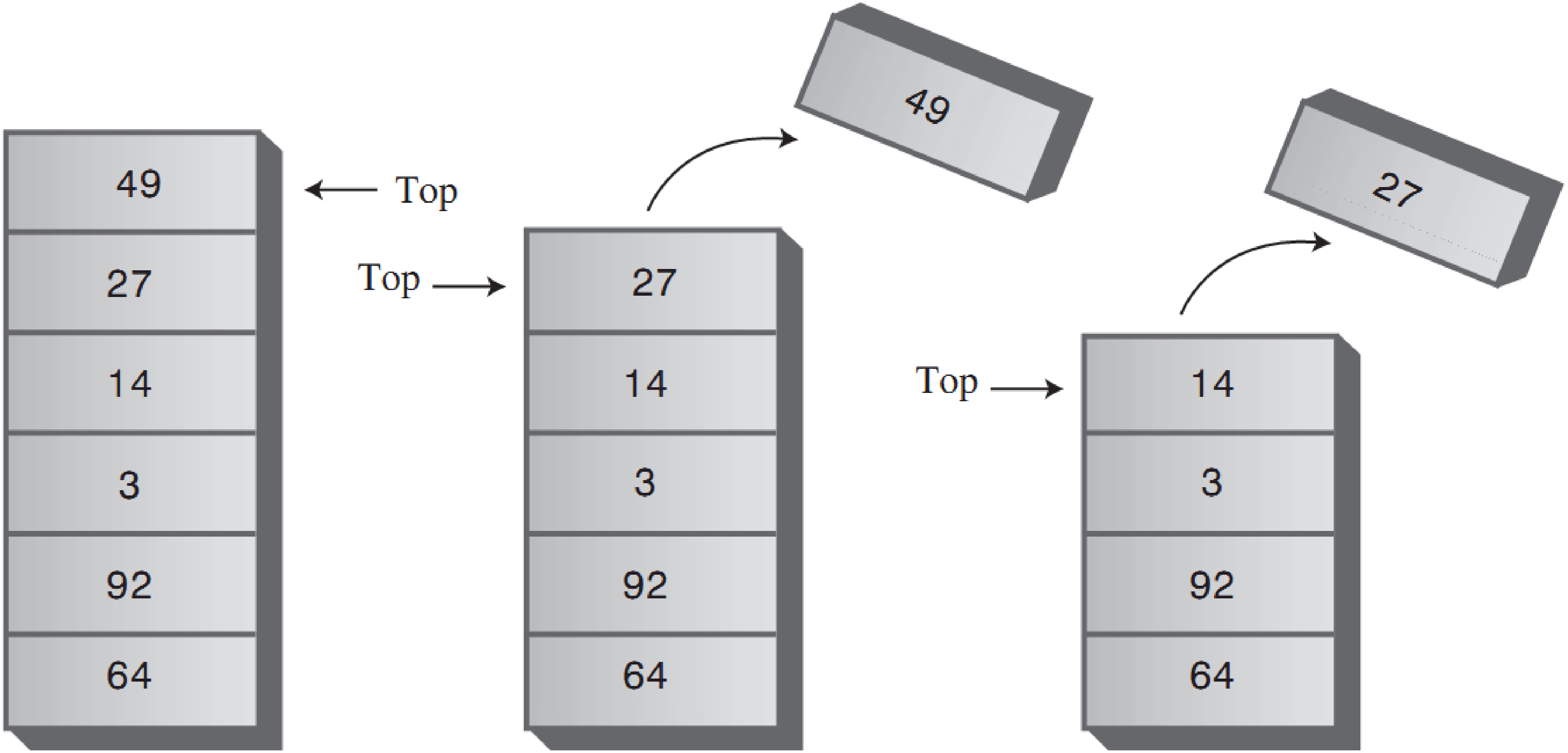
Operations

Push



Operations

Pop



Application

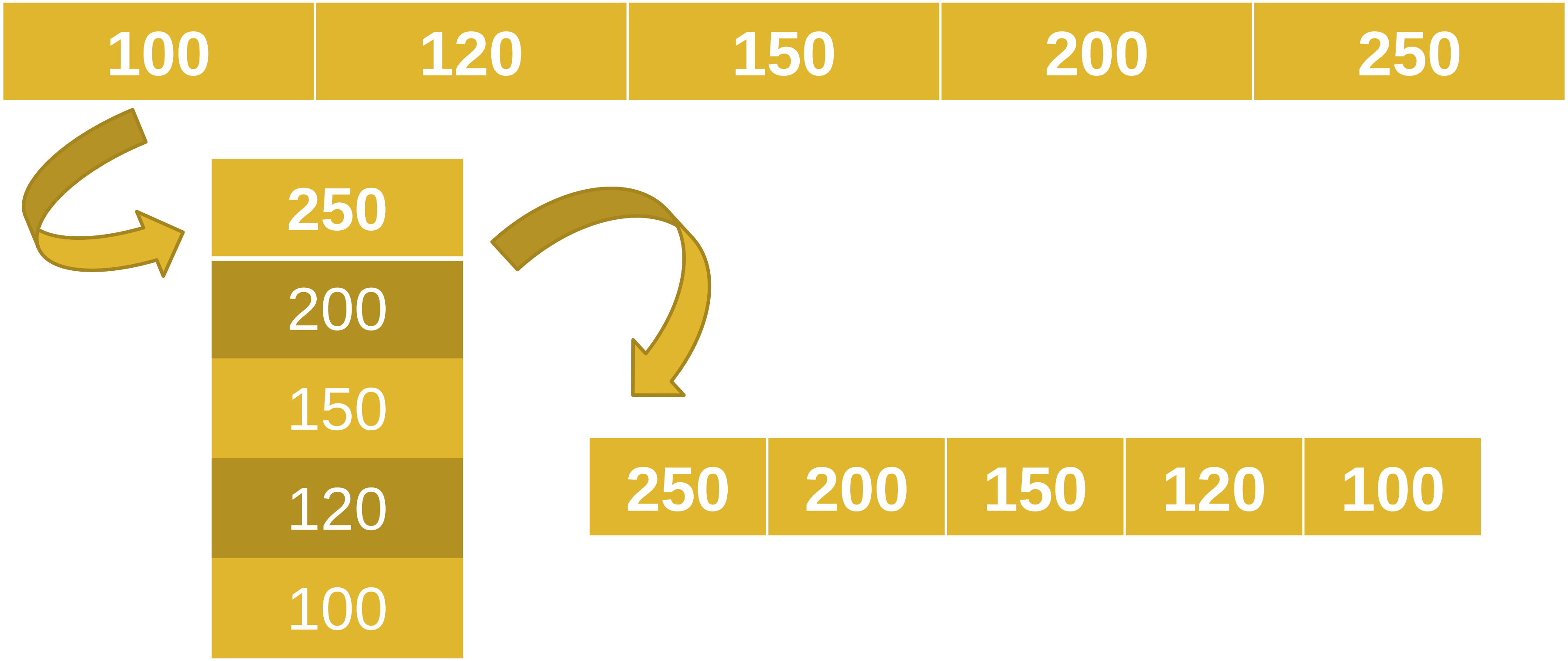
- Reversing an array/ a word

100	**120**	**150**	**200**	**250**

- Delimiter Matching
  - 100 * (100 – 50) → Correct
  - [100 * (100 – 50)] /2 → Correct
  - [100 * (100 – 50)} /2 → Incorrect, error on }
  - [100 * (100 – 50) /2 → Incorrect, error on [
  - (100 * (100 – 50))) /2 → Incorrect, error on)

How would you do it?

- Reversing a array



100	120	150	200	250

250
200
150
120
100

250	200	150	120	100

How would you do it?

- Delimiter Matching

(a * [b – c]) / d

Character Read	Stack contents
(	(
a	(
*	(
[	([
b	([
-	([
c	([
]	(
)	
/	
d	

Implementation

- See code in page 130

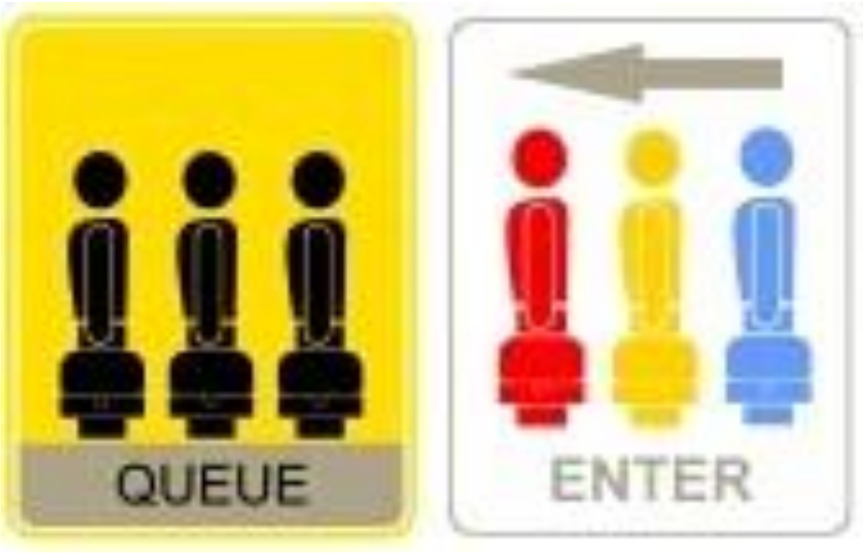
Efficiency of Stacks

- Complexity of
  - Push
  - Pop
  - Peek

  - → All of them are O(1)

Queue

Introduction



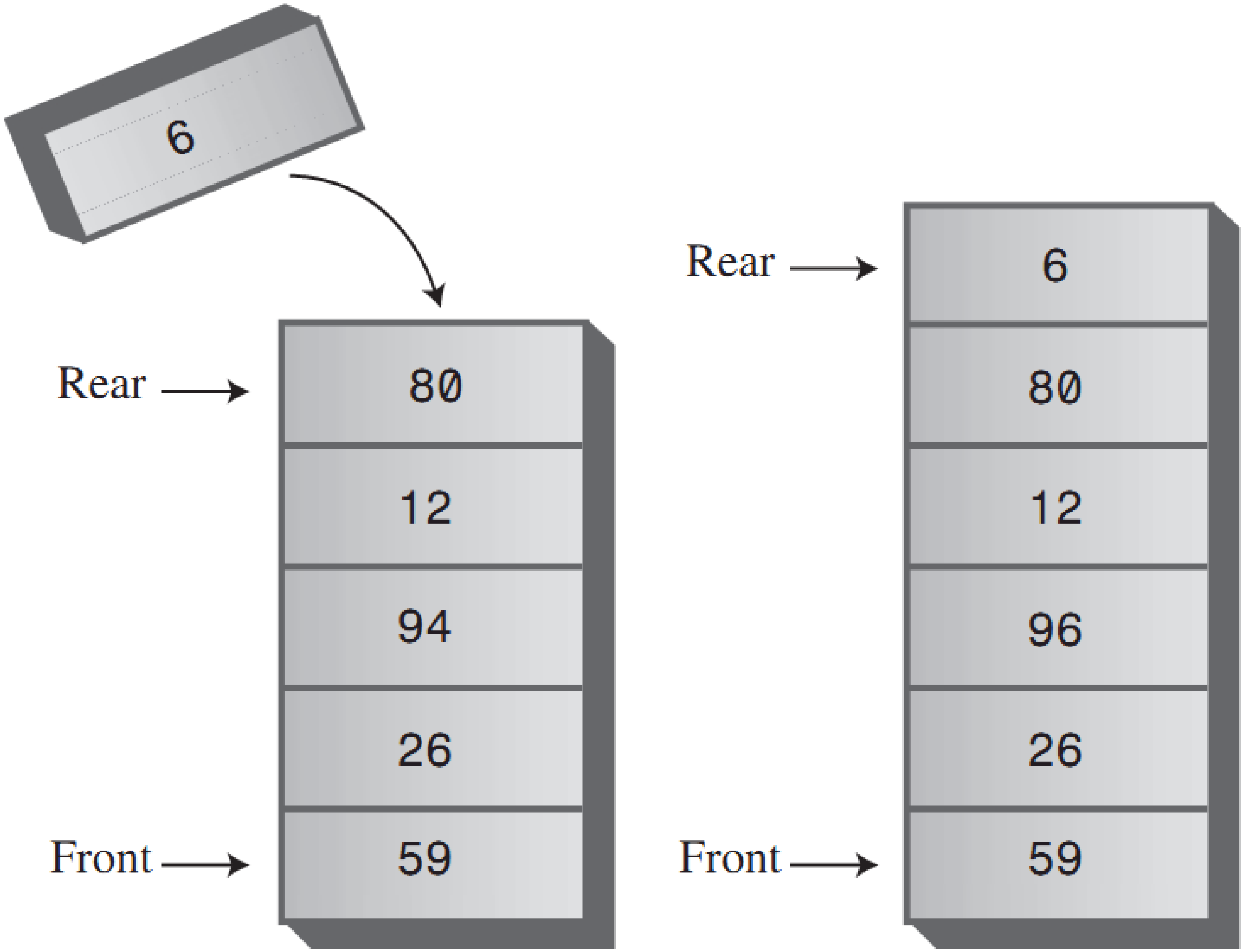
- Accessible item ?
  - → First inserted item
  - → First in, first out (FIFO)
- Tail / Head of queue
- Operations
  - Insert / Enqueue
  - Remove / Dequeue
- Properties
  - Full
  - Empty

Queue info

- Info must be managed?
  - Queue size (is full?)
  - Number of element (is empty?)
  - Head / tail item (accessible item)

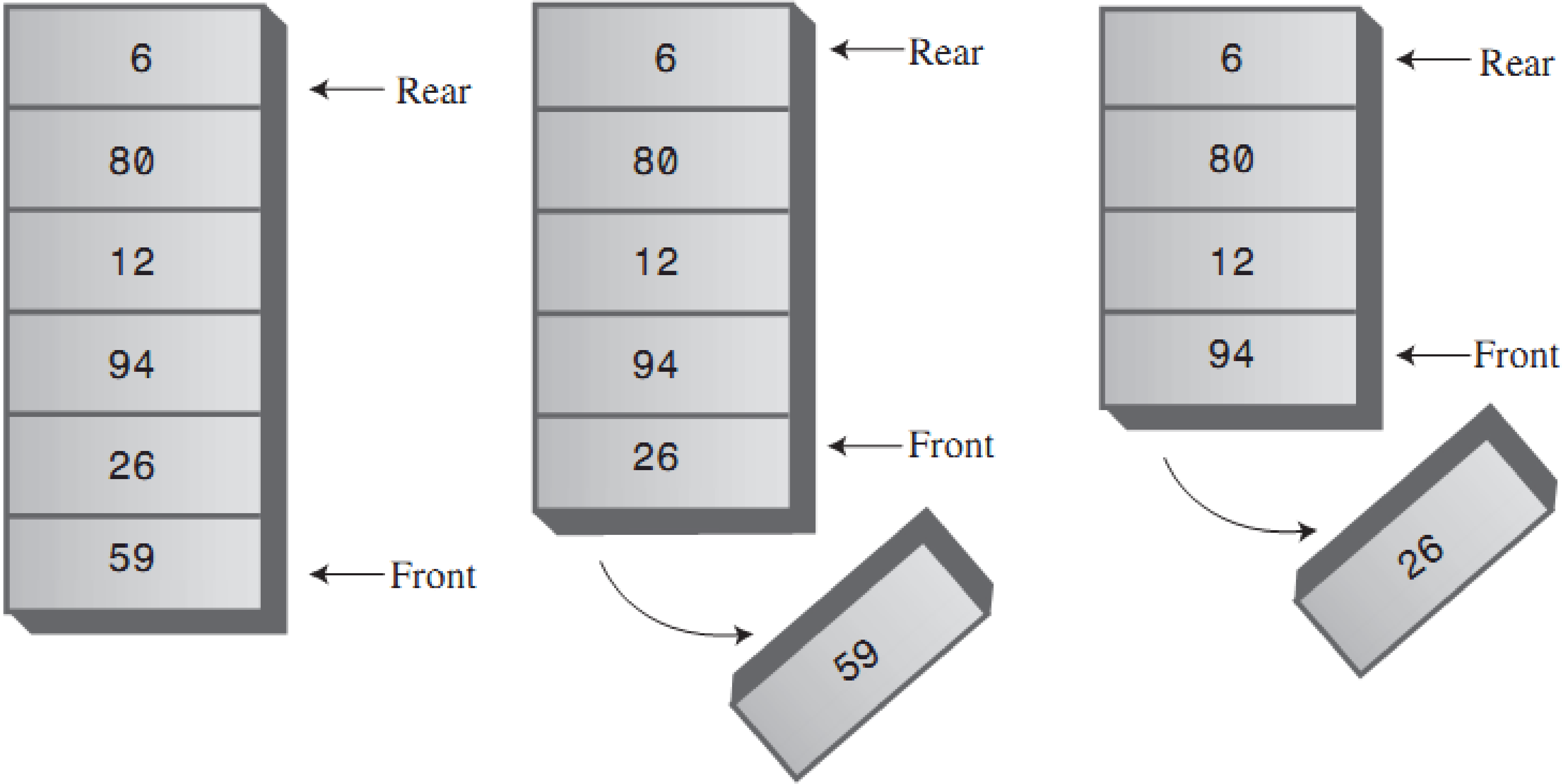
Operations

Enqueue



Operations

Dequeue

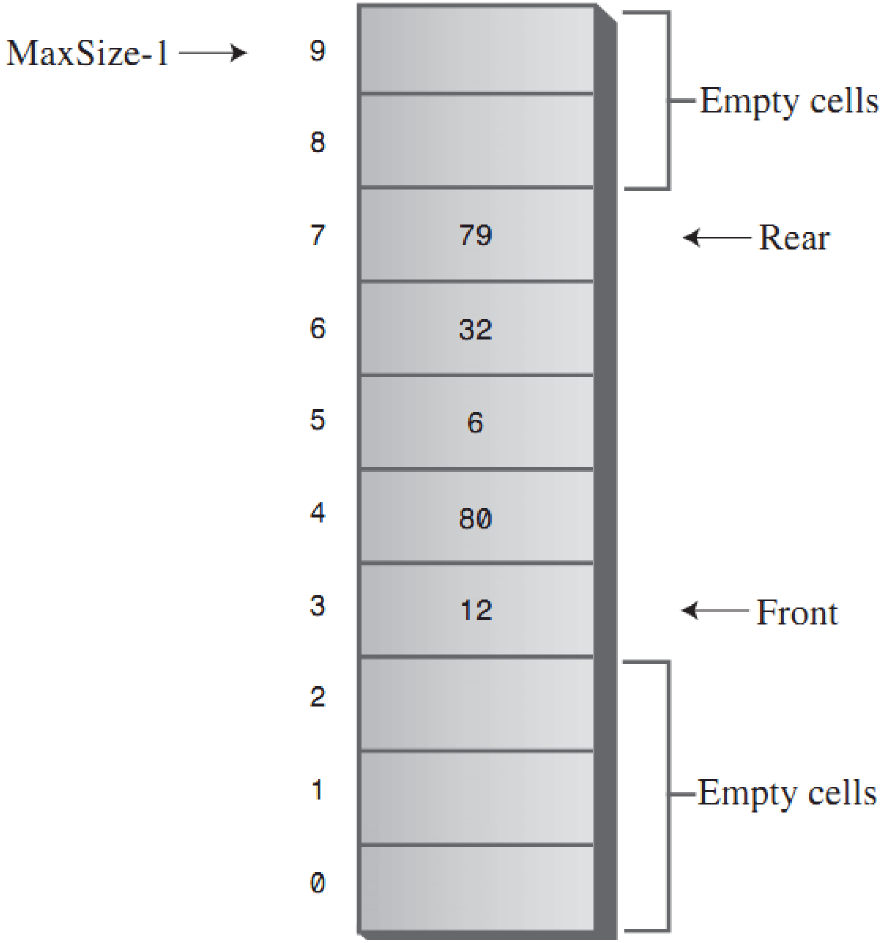


Application

- Printer queue
- File queue
- Request queue

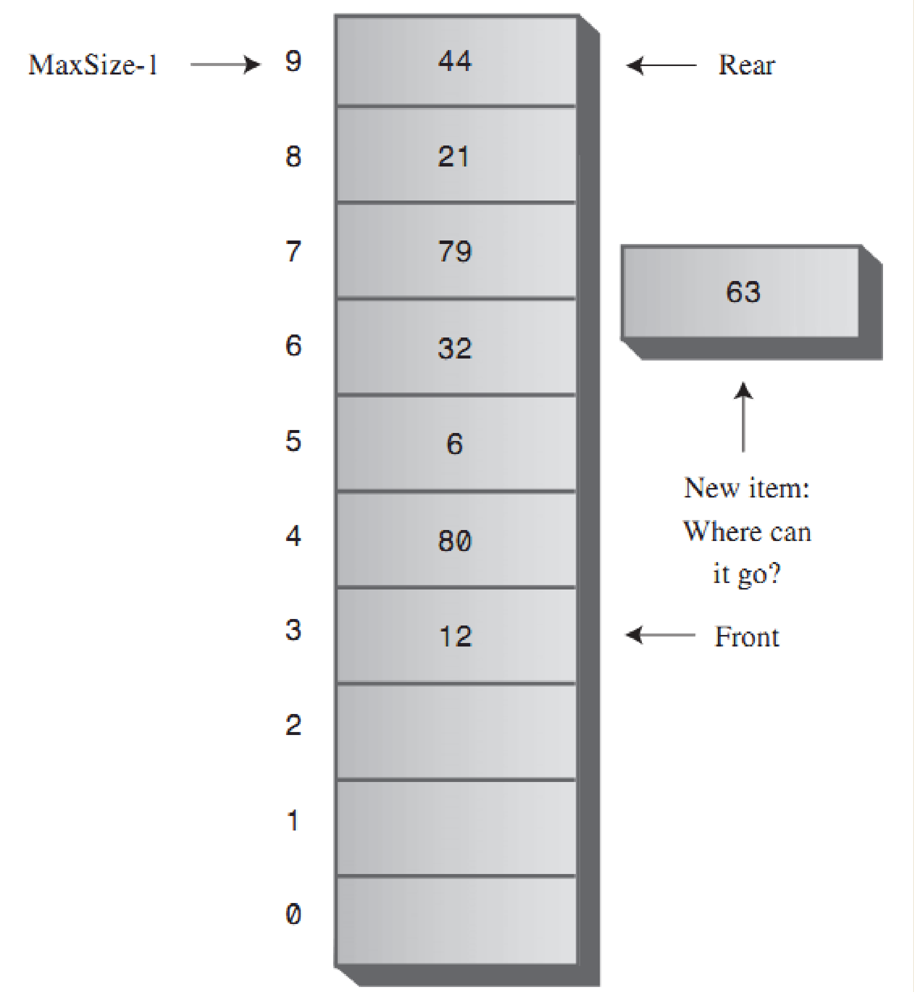
Implementation

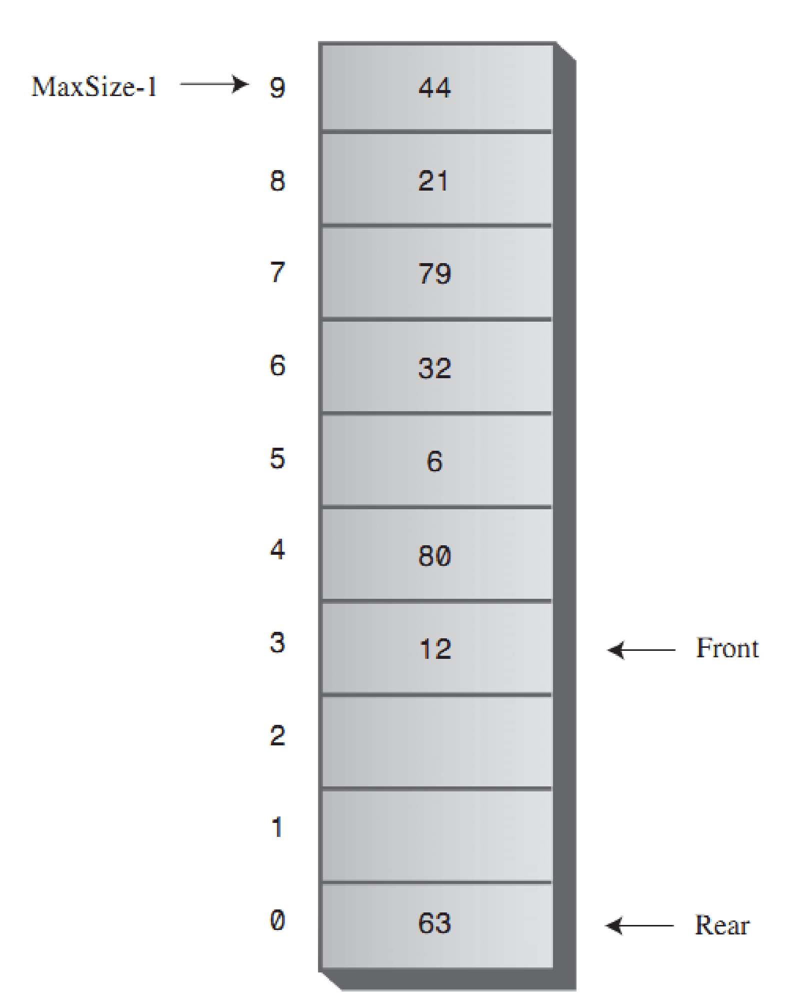
Empty cells



Implementation

Circular queue (Wrap around)





Look at some code

- Textbox – p.137

Efficiency of queue

- Enqueue?
- Dequeue?

→ O(1)

Question

Which of the following is true?

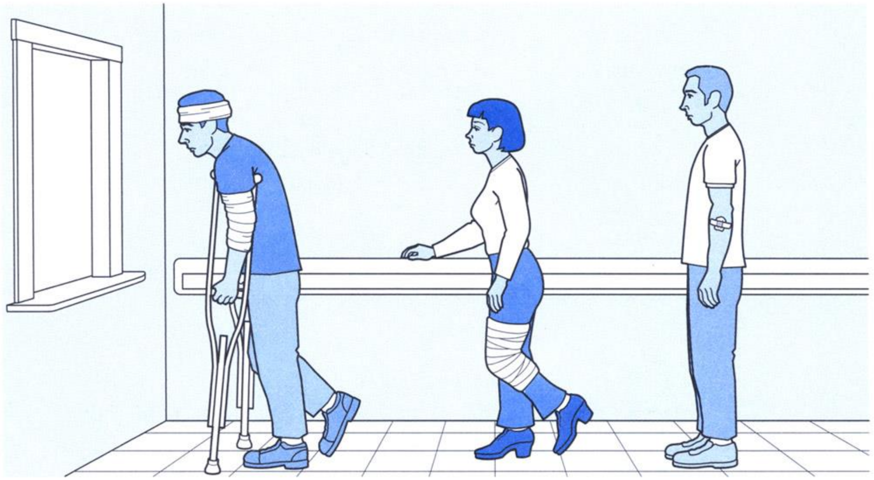
a. The pop operation on a stack is considerably simpler than the remove operation on a queue.

b. The contents of a queue can wrap around, while those of a stack cannot.

c. The top of a stack corresponds to the front of a queue.

d. In both the stack and the queue, items removed in sequence are taken from increasingly high index cells in the array.

Priority queue



- Head / tail
- Enqueue / Dequeue with criteria
- E.g, dequeue
  - Highest value, or
  - Most severe patient, …
- Ascending-priority / descending-priority queue

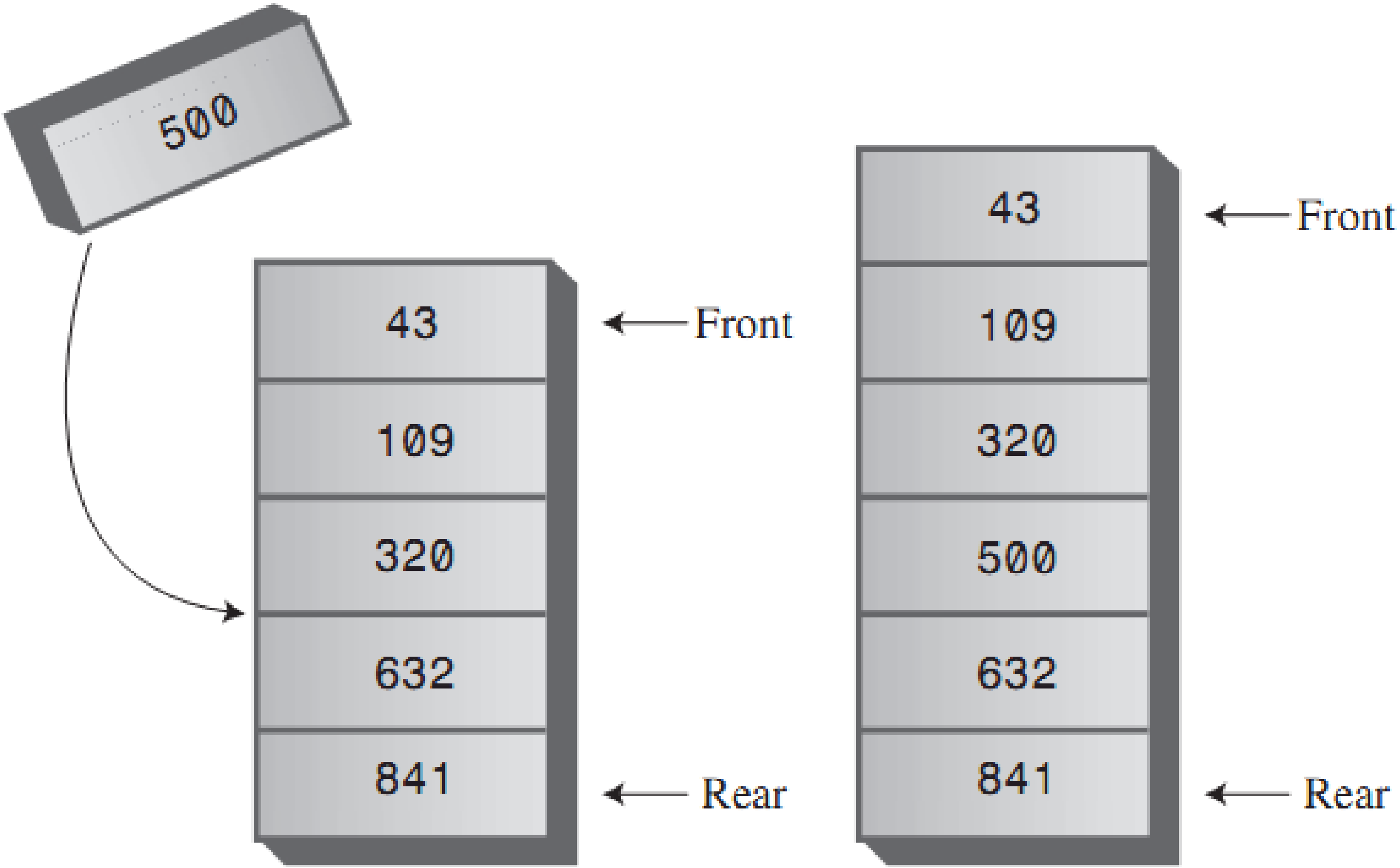
Efficiency of Priority queue

- Insertion ?
  - If use ARRAY: O(N)

- Deletion ?
  - O(1)

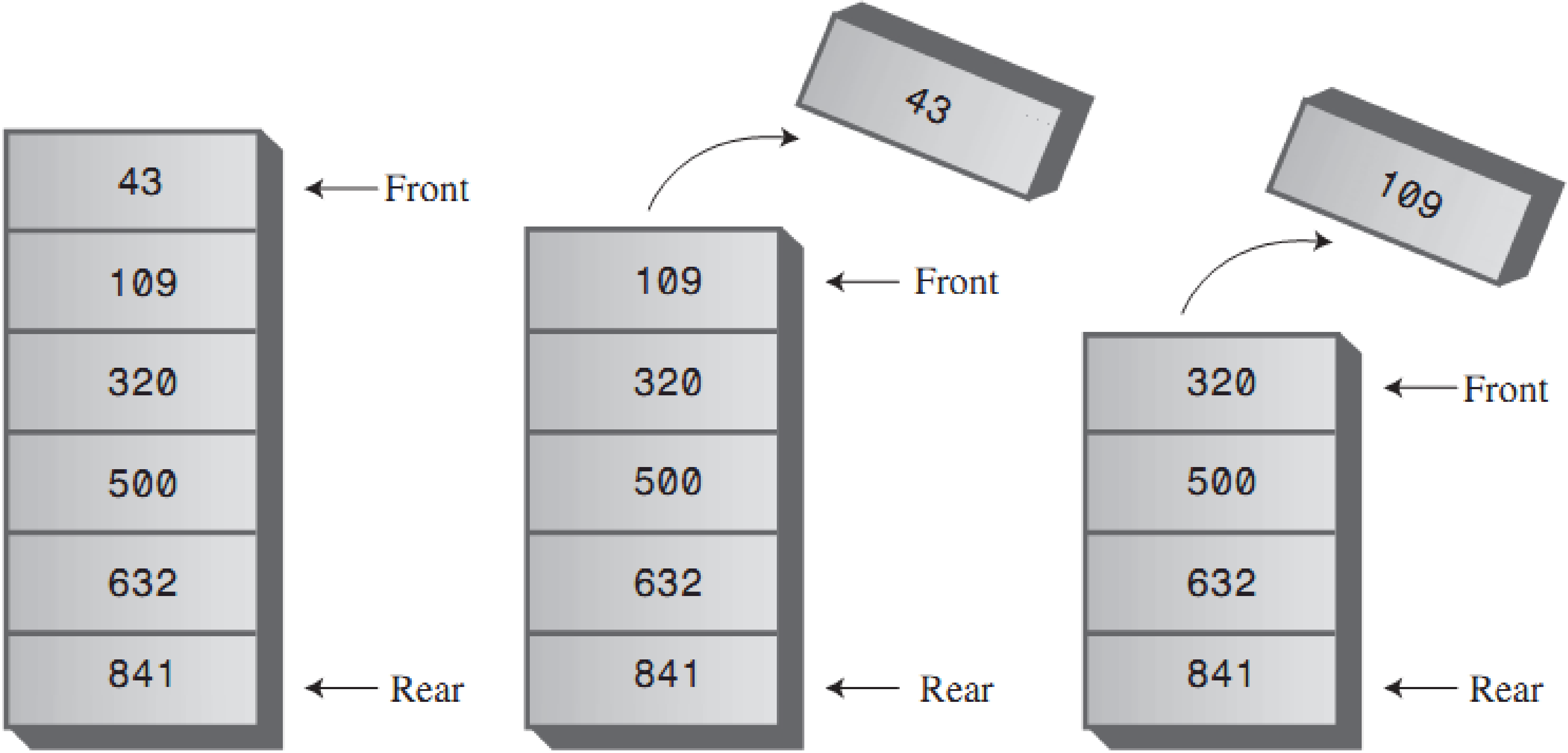
Operation

Enqueue



Operation

Dequeue



Question

One difference between a priority queue and an ordered array is that

a. the lowest-priority item cannot be extracted easily from the array as it can from the priority queue.

b. the array must be ordered while the priority queue need not be.

c. the highest priority item can be extracted easily from the priority queue but not from the array.

d. All of the above.

Parsing Arithmetic Expression

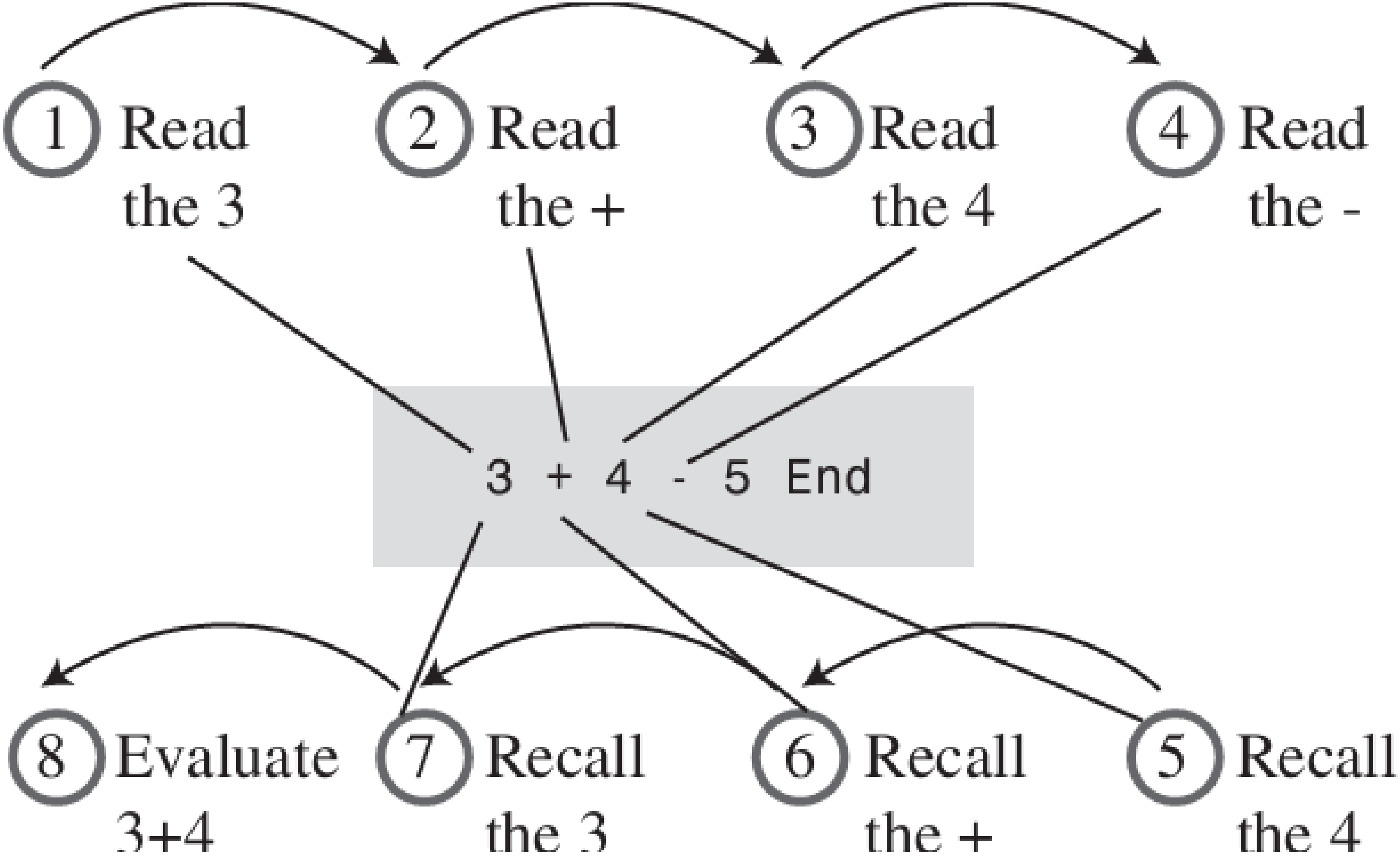
Introduction

- How would you evaluate an expression?

3 + 4 - 5

Or (10-5)*2 + (6+4)*3

Evaluate 3 + 4 - 5



1 Read the 3

2 Read the +

3 Read the 4

4 Read the -

3 + 4 - 5 End

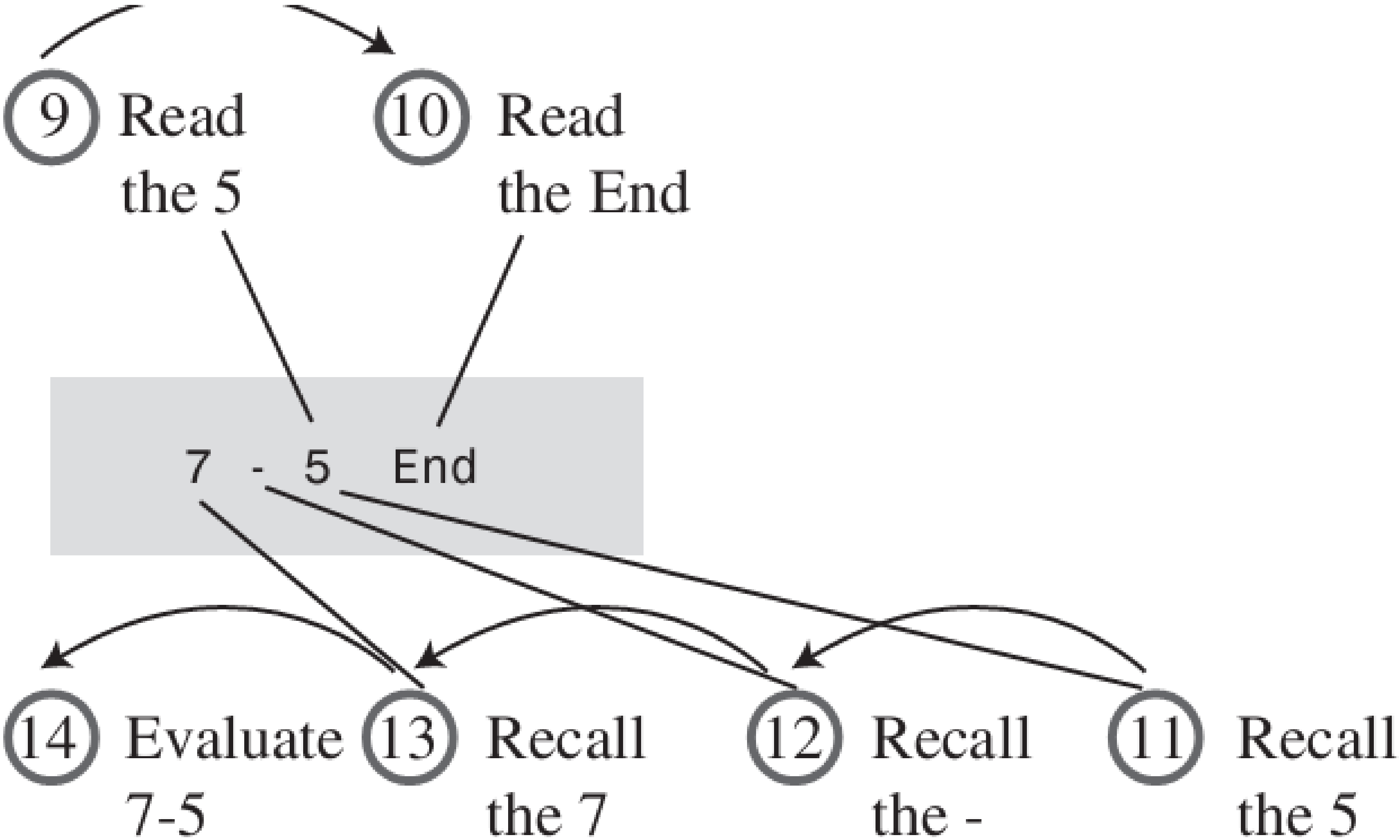
5 Recall the 4

6 Recall the +

7 Recall the 3

8 Evaluate 3+4

Evaluate 3 + 4 - 5



Algorithm

- For computer algorithms:

difficult to evaluate arithmetic expression directly

Solution

- Transform arithmetic expression into a different format – POSTFIX
- Evaluation the postfix expression

Postfix Notation

- To develop a string where the operators (*, -, +,…) appear ‘last’ (hence the term: postfix).
  - e.g. ab+.
  - Also known as Reverse Polish Notation
- Normally use infix notation
  - a+b.
  - Most of the operators we use are binary.
- There is also a ‘prefix’ notation, which has more limited applications.

Infix and postfix notations

- In postfix notation, an operator operates on the two previous operands. That is the rule.

Table: infix to postfix notations

- Parentheses override normal hierarchical evaluation
-

Infix	Postfix
a+b-c	ab+c-
a*b/c	ab*c/
a+b*c	abc*+
a*b+c	ab*c+
a*(b+c)	abc+*
a*b+c*d	ab*cd*+
((a+b)*c)-d	ab+c*d-
a+b*(c-d/(e+f))	abcdef+/-*+

How Humans Translate Infix into Postfix

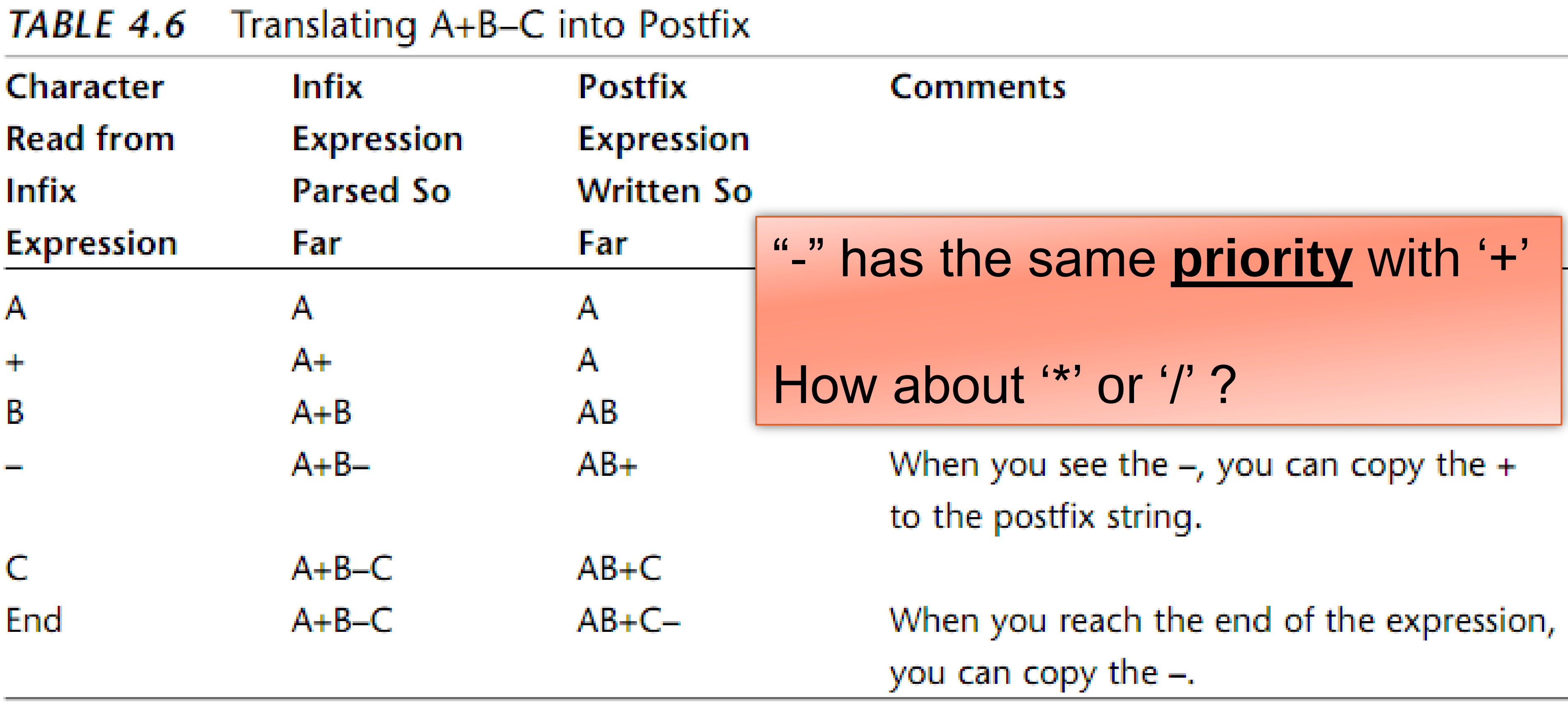


TABLE 4.6 Translating A+B–C into Postfix

Character Read from Infix Expression	Infix Expression Parsed So Far	Postfix Expression Written So Far	Comments
A	A	A	
+	A+	A	
B	A+B	AB	
–	A+B–	AB+	When you see the –, you can copy the + to the postfix string.
C	A+B–C	AB+C	
End	A+B–C	AB+C–	When you reach the end of the expression, you can copy the –.

“-” has the same **priority** with ‘+’

How about ‘*’ or ‘/’ ?

How Humans Translate Infix into Postfix

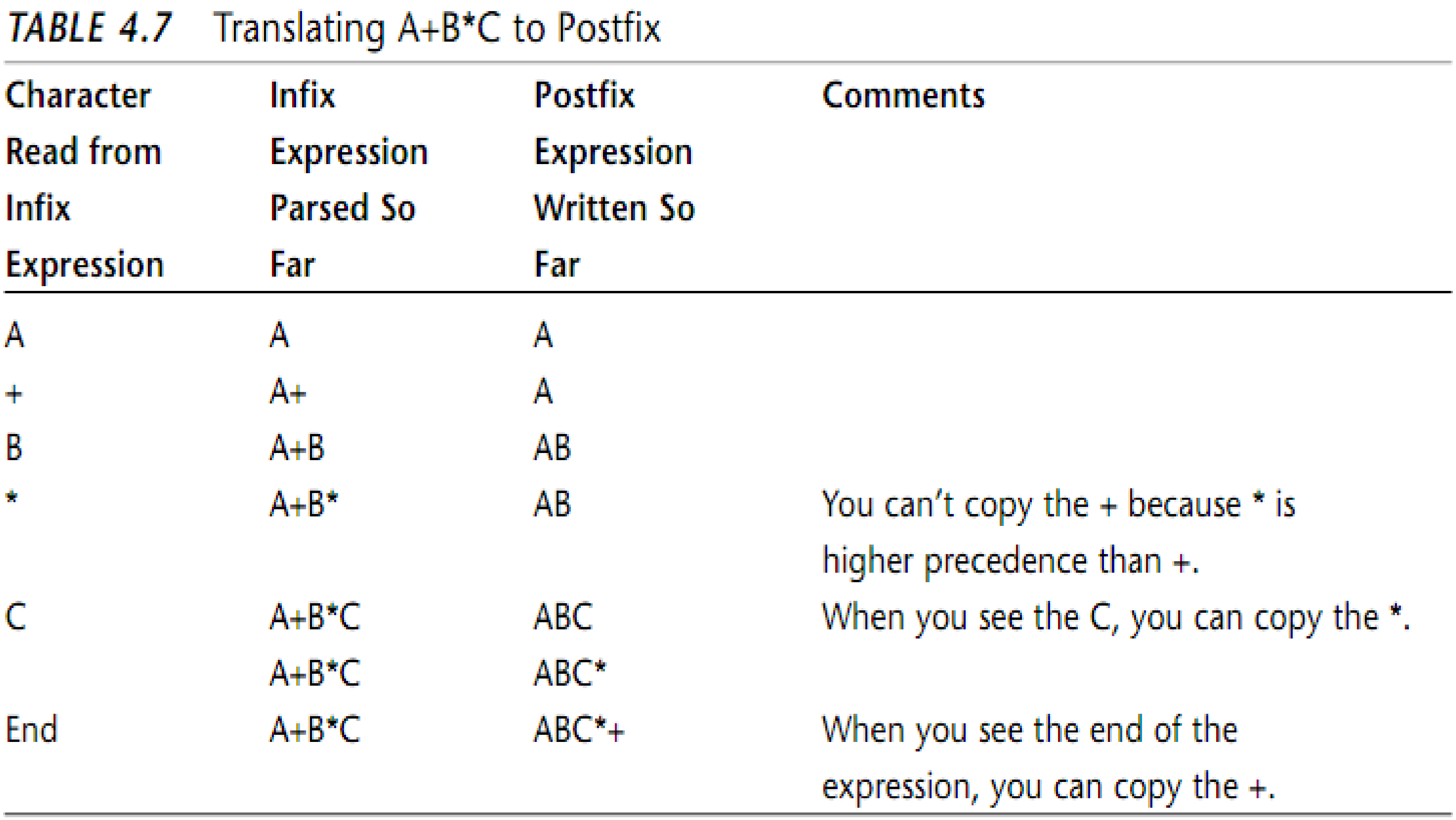


TABLE 4.7 Translating A+B*C to Postfix

Character Read from Infix Expression	Infix Expression Parsed So Far	Postfix Expression Written So Far	Comments
A	A	A	
+	A+	A	
B	A+B	AB	
*	A+B*	AB	You can't copy the + because * is higher precedence than +.
C	A+B*C	ABC	When you see the C, you can copy the *.
	A+B*C	ABC*	
End	A+B*C	ABC*+	When you see the end of the expression, you can copy the +.

How Humans Translate Infix into Postfix

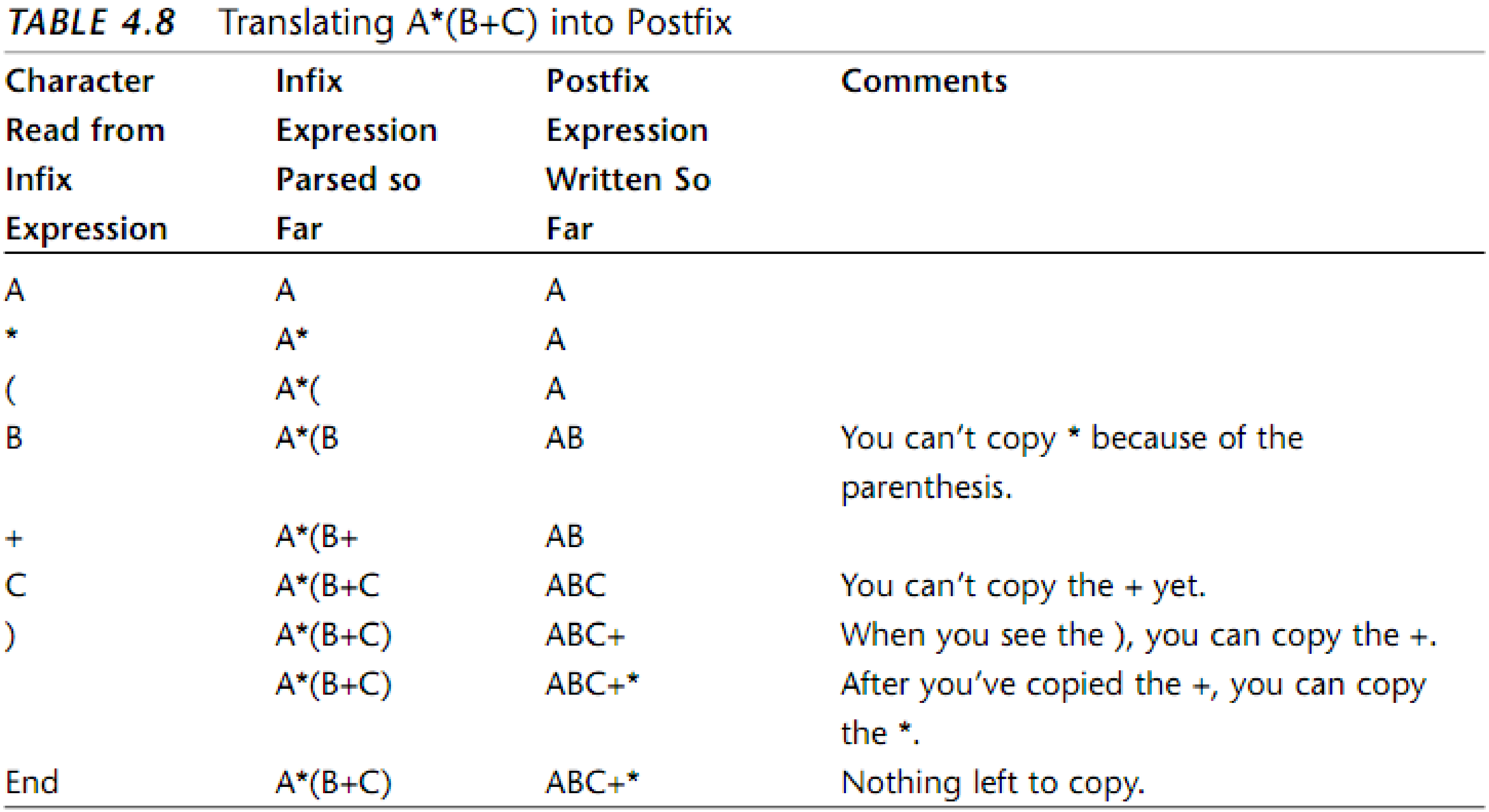


TABLE 4.8 Translating A*(B+C) into Postfix

Character Read from Infix Expression	Infix Expression Parsed so Far	Postfix Expression Written So Far	Comments
A	A	A	
*	A*	A	
(	A*(	A	
B	A*(B	AB	You can't copy * because of the parenthesis.
+	A*(B+	AB	
C	A*(B+C	ABC	You can't copy the + yet.
)	A*(B+C)	ABC+	When you see the), you can copy the +.
	A*(B+C)	ABC+*	After you've copied the +, you can copy the *.
End	A*(B+C)	ABC+*	Nothing left to copy.

How Humans Translate Infix into Postfix

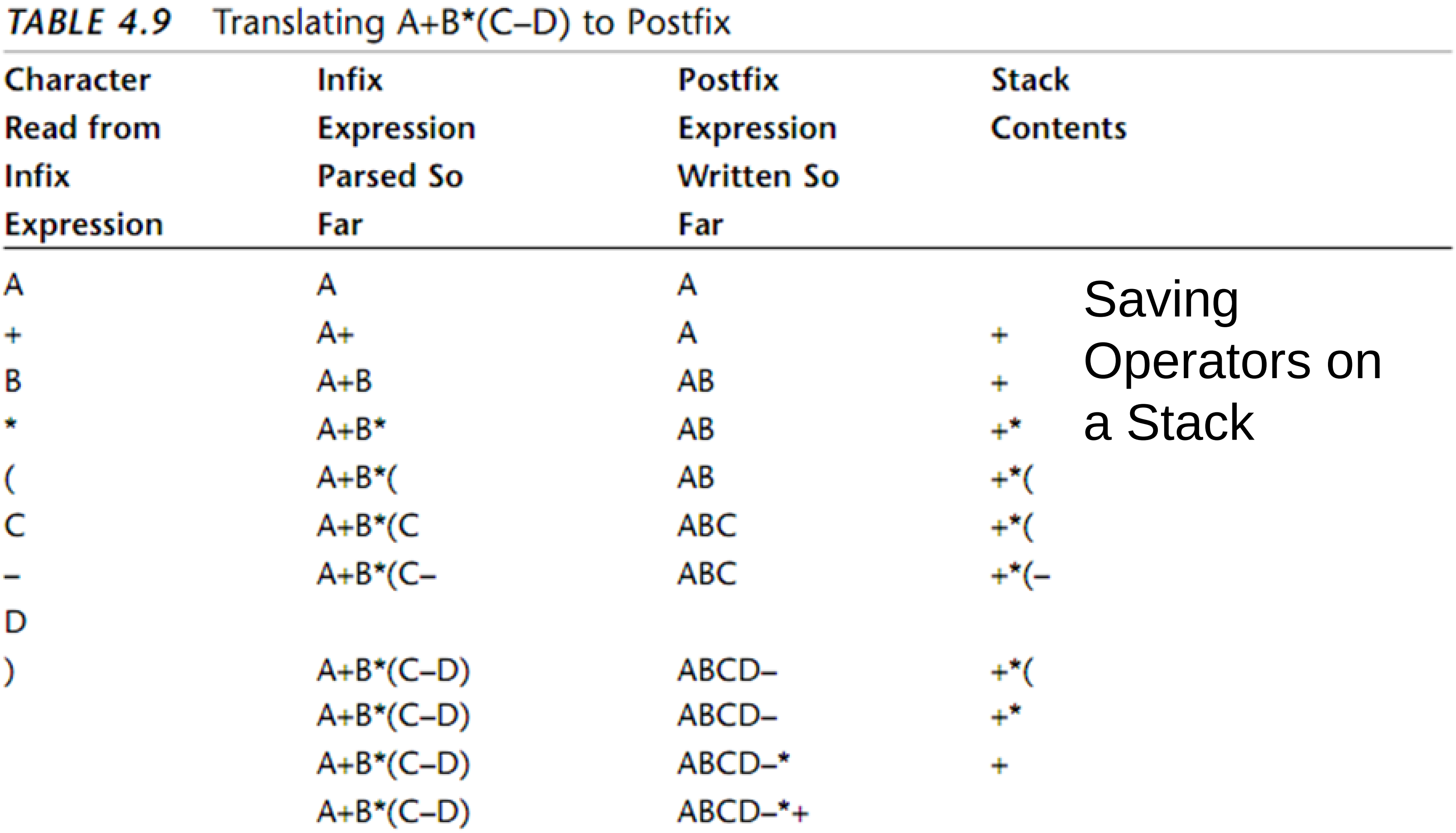


TABLE 4.9 Translating A+B*(C–D) to Postfix

Character Read from Infix Expression	Infix Expression Parsed So Far	Postfix Expression Written So Far	Stack Contents
A	A	A	
+	A+	A	+
B	A+B	AB	+
*	A+B*	AB	+*
(	A+B*(	AB	+*(
C	A+B*(C	ABC	+*(
–	A+B*(C–	ABC	+*(–
D			
)	A+B*(C–D)	ABCD–	+*(
	A+B*(C–D)	ABCD–	+*
	A+B*(C–D)	ABCD–*	+
	A+B*(C–D)	ABCD–*+	

Saving Operators on a Stack

TABLE 4.10 Infix to Postfix Translation Rules

Item Read from Input (Infix)	**Action**
Operand	Write it to output (postfix)
Open parenthesis (	Push it on stack
Close parenthesis)	While stack not empty, repeat the following: Pop an item, If item is not (, write it to output Quit loop if item is (

Operator (`opThis`)	If stack empty,
	  Push `opThis`
	Otherwise,
	  While stack not empty, repeat:
	    Pop an item,
	    If item is (, push it, or
	    If item is an operator (`opTop`), and
	      If `opTop < opThis`, push `opTop`, or
	      If `opTop >= opThis`, output `opTop`
	  Quit loop if `opTop < opThis` or item is (
	  Push `opThis`

No more items	While stack not empty, Pop item, output it.

Example

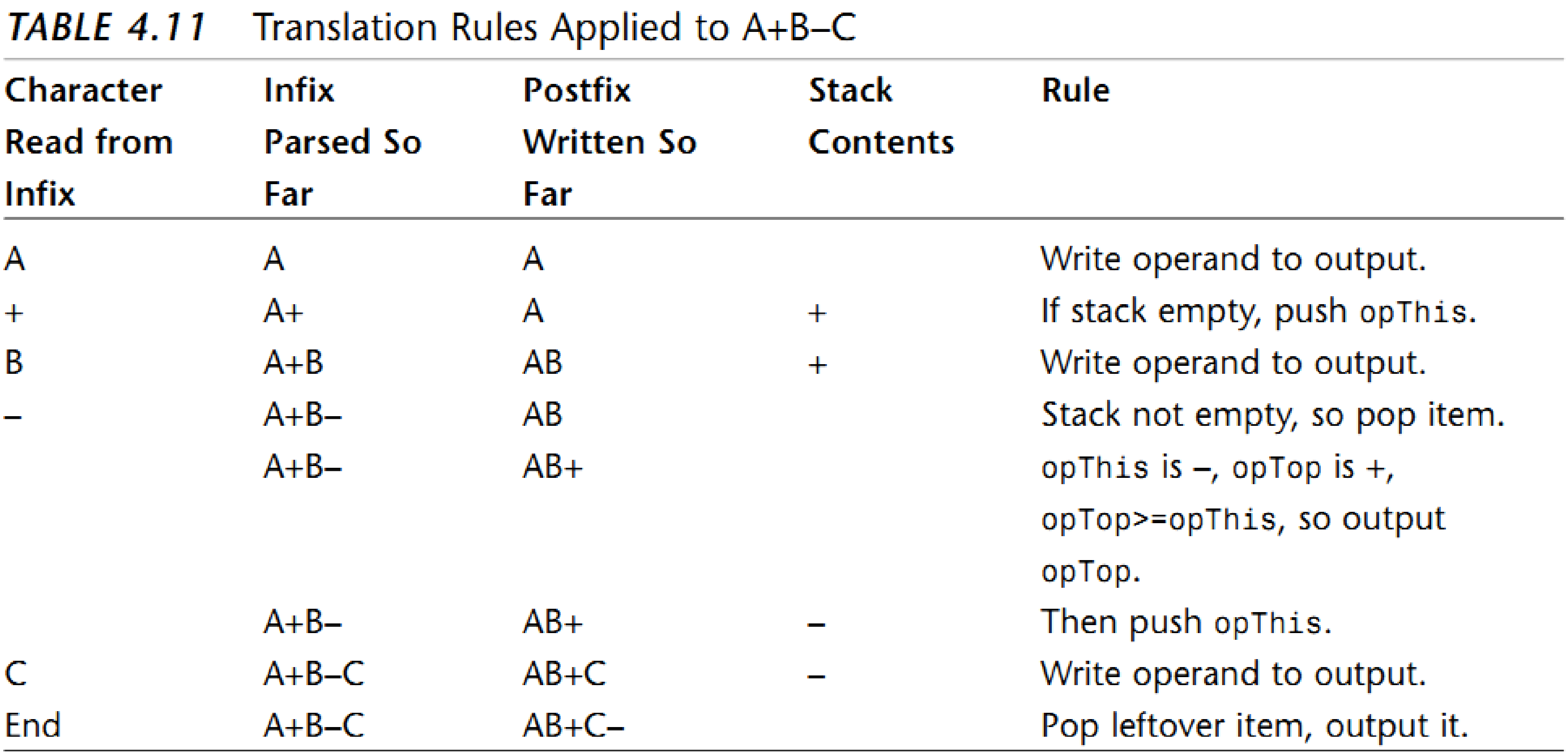


TABLE 4.11 Translation Rules Applied to A+B–C

Character Read from Infix	Infix Parsed So Far	Postfix Written So Far	Stack Contents	Rule
A	A	A		Write operand to output.
+	A+	A	+	If stack empty, push `opThis`.
B	A+B	AB	+	Write operand to output.
–	A+B–	AB		Stack not empty, so pop item.
	A+B–	AB+		`opThis` is –, `opTop` is +, `opTop>=opThis`, so output `opTop`.
	A+B–	AB+	–	Then push `opThis`.
C	A+B–C	AB+C	–	Write operand to output.
End	A+B–C	AB+C–		Pop leftover item, output it.

Example

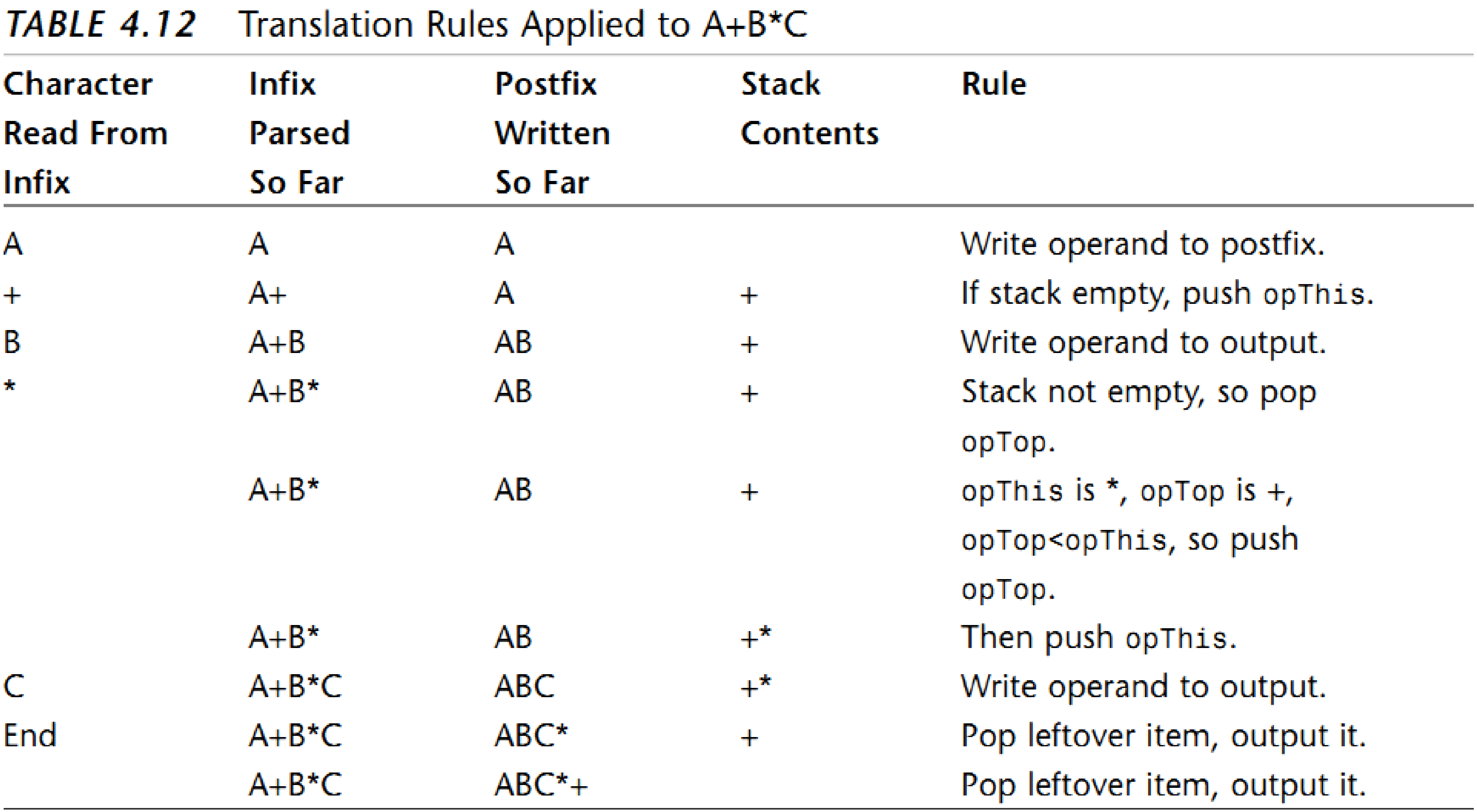


TABLE 4.12 Translation Rules Applied to A+B*C

Character	Infix	Postfix	Stack	Rule
Read From	Parsed	Written	Contents	
Infix	So Far	So Far		
A	A	A		Write operand to postfix.
+	A+	A	+	If stack empty, push `opThis`.
B	A+B	AB	+	Write operand to output.
*	A+B*	AB	+	Stack not empty, so pop `opTop`.
	A+B*	AB	+	`opThis` is *, `opTop` is +, `opTop<opThis`, so push `opTop`.
	A+B*	AB	+*	Then push `opThis`.
C	A+B*C	ABC	+*	Write operand to output.
End	A+B*C	ABC*	+	Pop leftover item, output it.
	A+B*C	ABC*+		Pop leftover item, output it.

Example

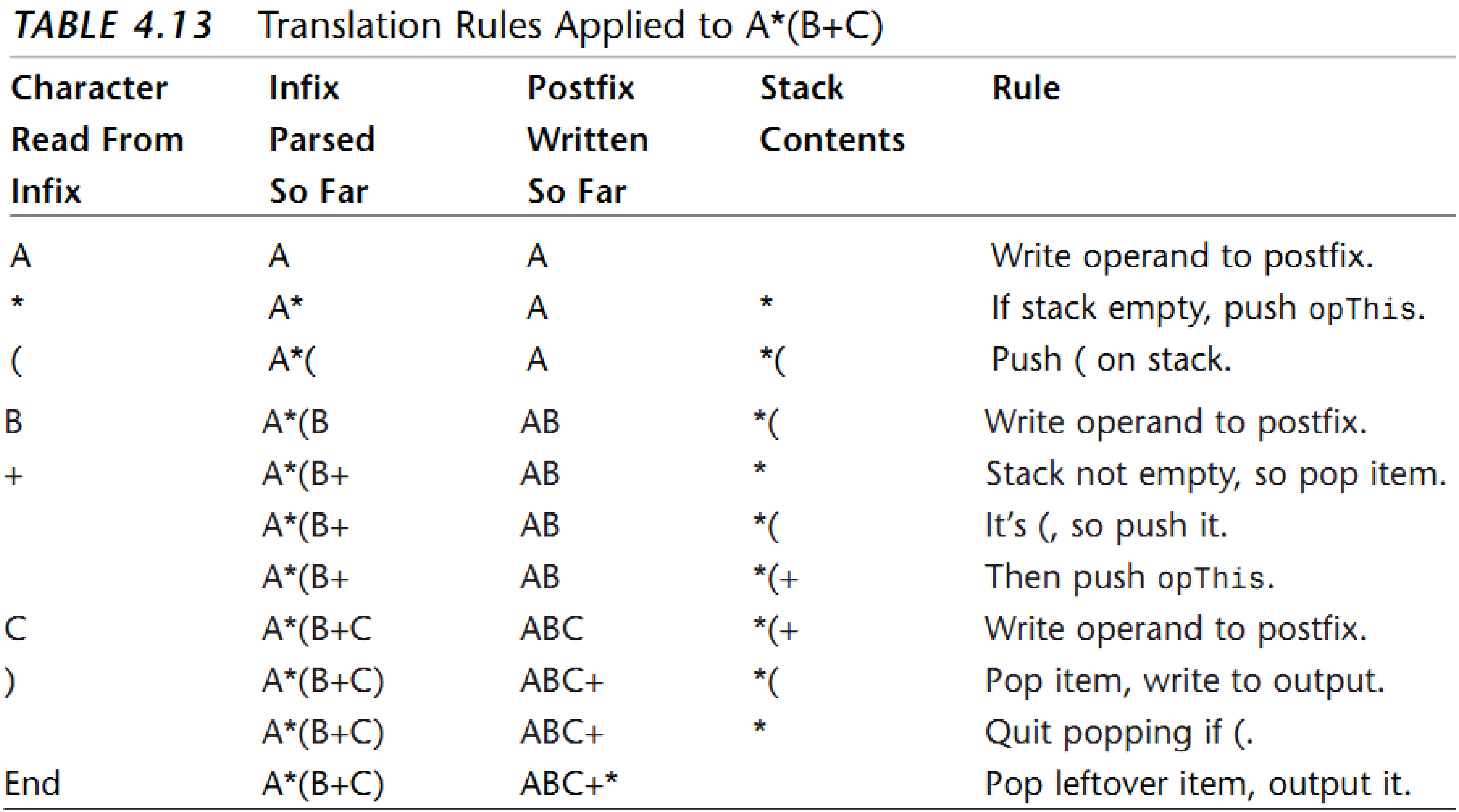


TABLE 4.13 Translation Rules Applied to A*(B+C)

Character Read From Infix	Infix Parsed So Far	Postfix Written So Far	Stack Contents	Rule
A	A	A		Write operand to postfix.
*	A*	A	*	If stack empty, push `opThis`.
(	A*(	A	*(	Push (on stack.
B	A*(B	AB	*(	Write operand to postfix.
+	A*(B+	AB	*	Stack not empty, so pop item.
	A*(B+	AB	*(	It's (, so push it.
	A*(B+	AB	*(+	Then push `opThis`.
C	A*(B+C	ABC	*(+	Write operand to postfix.
)	A*(B+C)	ABC+	*(	Pop item, write to output.
	A*(B+C)	ABC+	*	Quit popping if (.
End	A*(B+C)	ABC+*		Pop leftover item, output it.