DATA STRUCTURE ALGORITHMS AND APPLICATIONS (CT- 159)

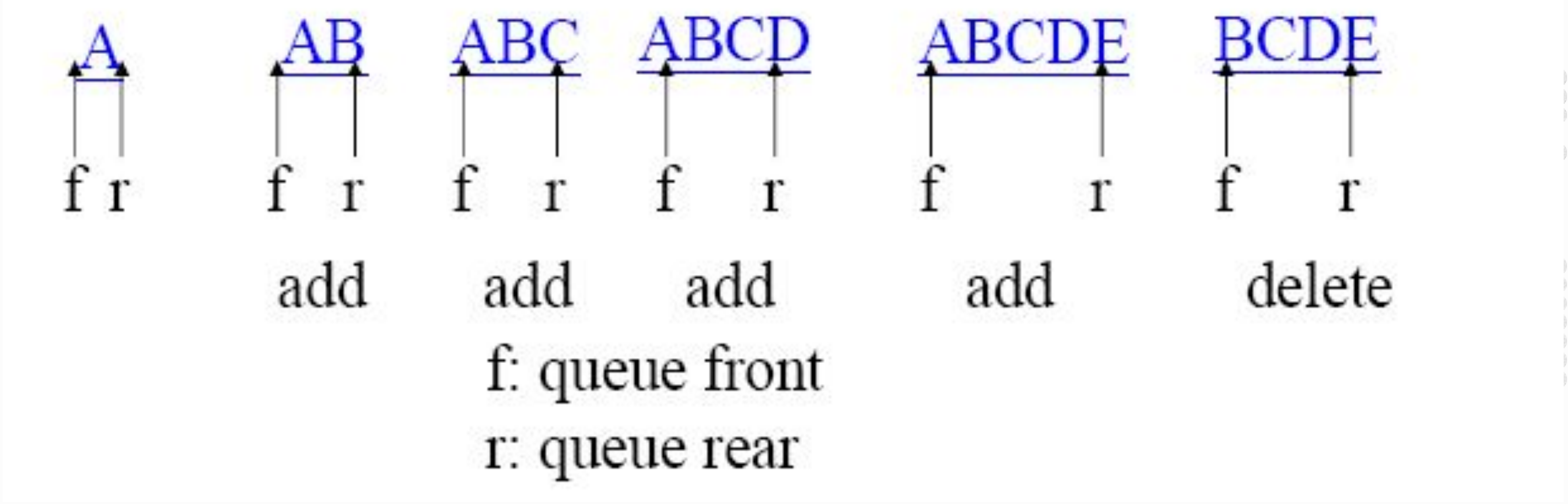
Lecture – 4
Elementary Sorting Techniques: Bubble Sort, Insertion Sort, Selection Sort

Instructor: Engr. Nasr Kamal

Department of Computer Science and Information Technology

Queue

- A queue is also known as a First-In-First-Out (FIFO) list.
- A queue is a linear list of elements in which deletions can take place only at one end, called the front, and insertion can take place only at the other end called the rear.



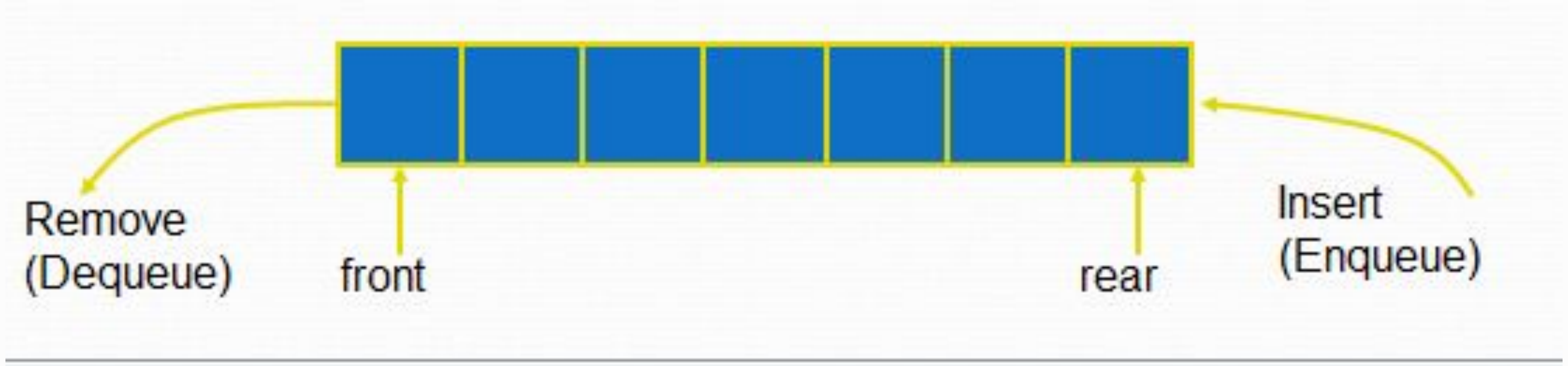
Queue Operations

- The queue supports three fundamental methods:
- new():ADT-Creates an empty queue..
- Enqueue(S:ADT,O:element):ADT-Inserts object O at the rear of the queue.
- Dequeue(S:ADT):ADT-Removes the object from the front of the queue; an error occurs if the queue is empty.
- Front(S:ADT):element-Returns, but does not remove, the front element; an error occurs if the queue is empty.
- These support methods should also be defines:
- Size(S:ADT):integer
- IsEmpty(S:ADT):boolean

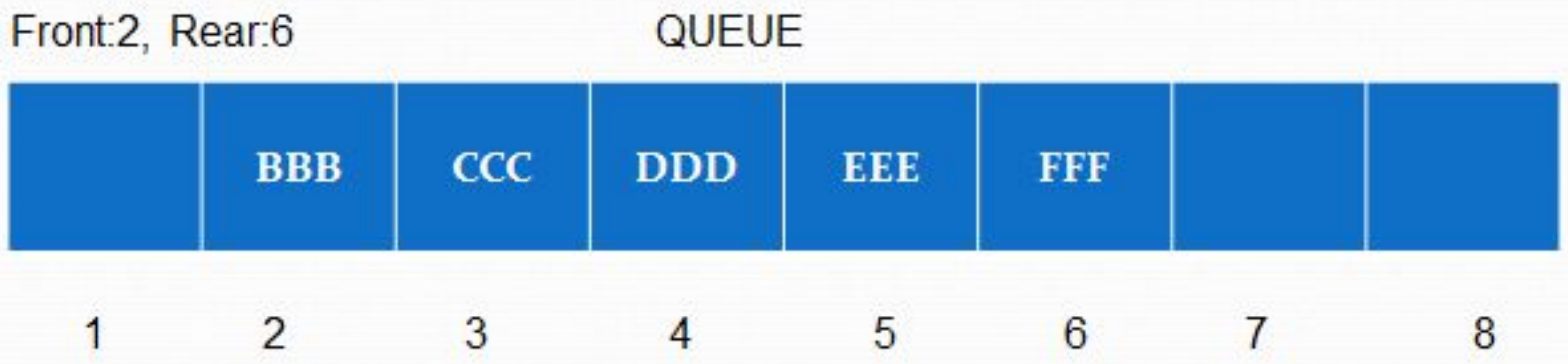
Enqueue and Dequeue

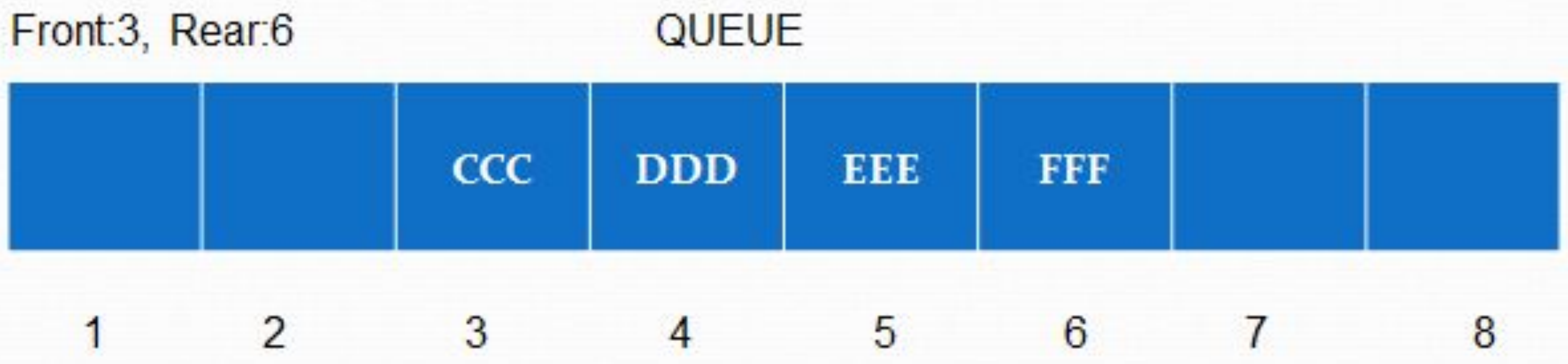
Primary queue operations: Enqueue and Dequeue
Like check-out lines in a store, a queue has a front and a rear.

- Enqueue: Insert an element at the rear of the queue
- Dequeue: Remove an element from the front of the queue



Implementing Queue using Array





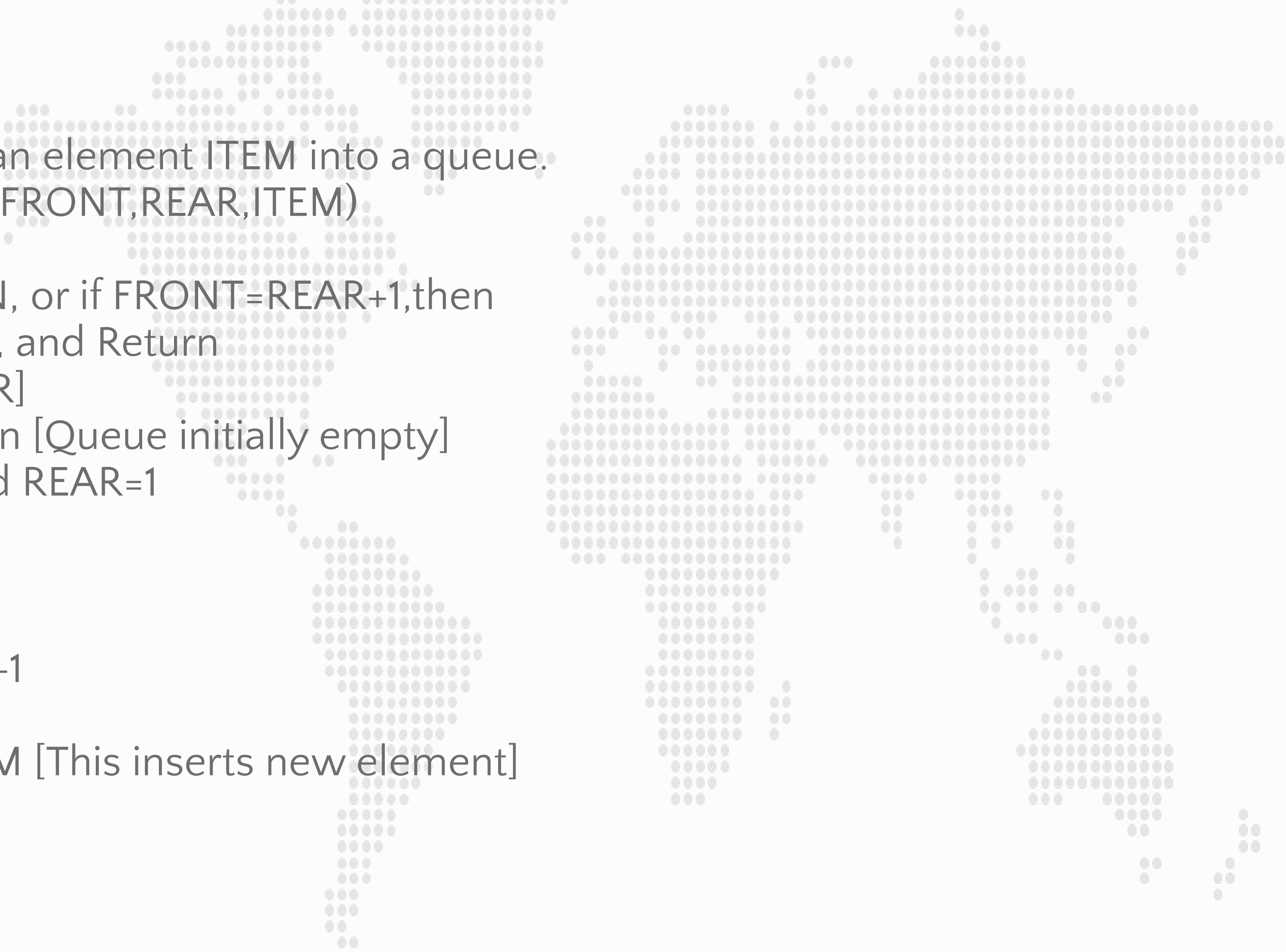
Circular Queue

- Suppose we want to insert an element into the queue after it occupies the last part of the array i.e. REAR=n.

- One way of doing it is simply move the entire queue to the beginning of the array, changing FRONT and REAR, and then inserting the element. This procedure may be very expensive.

- This can be done assuming that the array QUEUE is circular that is the QUEUE[1] comes after QUEUE[N] in the array. Instead if increasing REAR to N+1 we reset REAR=1.

QInsert



This procedure inserts an element ITEM into a queue.

```
    QINSERT(QUEUE,N,FRONT,REAR,ITEM)
    [Queue already fill]
If FRONT=1 and REAR=N, or if FRONT=REAR+1,then
        Write: Overflow, and Return
[Find new value of REAR]
    If FRONT=NULL, then [Queue initially empty]
        Set FRONT=1 and REAR=1
    Else if REAR =N then
        Set REAR=1
    Else
        Set REAR=REAR+1
[End of if structure]
Set  QUEUE[REAR]=ITEM [This inserts new element]
Return
```

QDelete

This procedure deletes an element from the queue and assigns it to the variable ITEM.

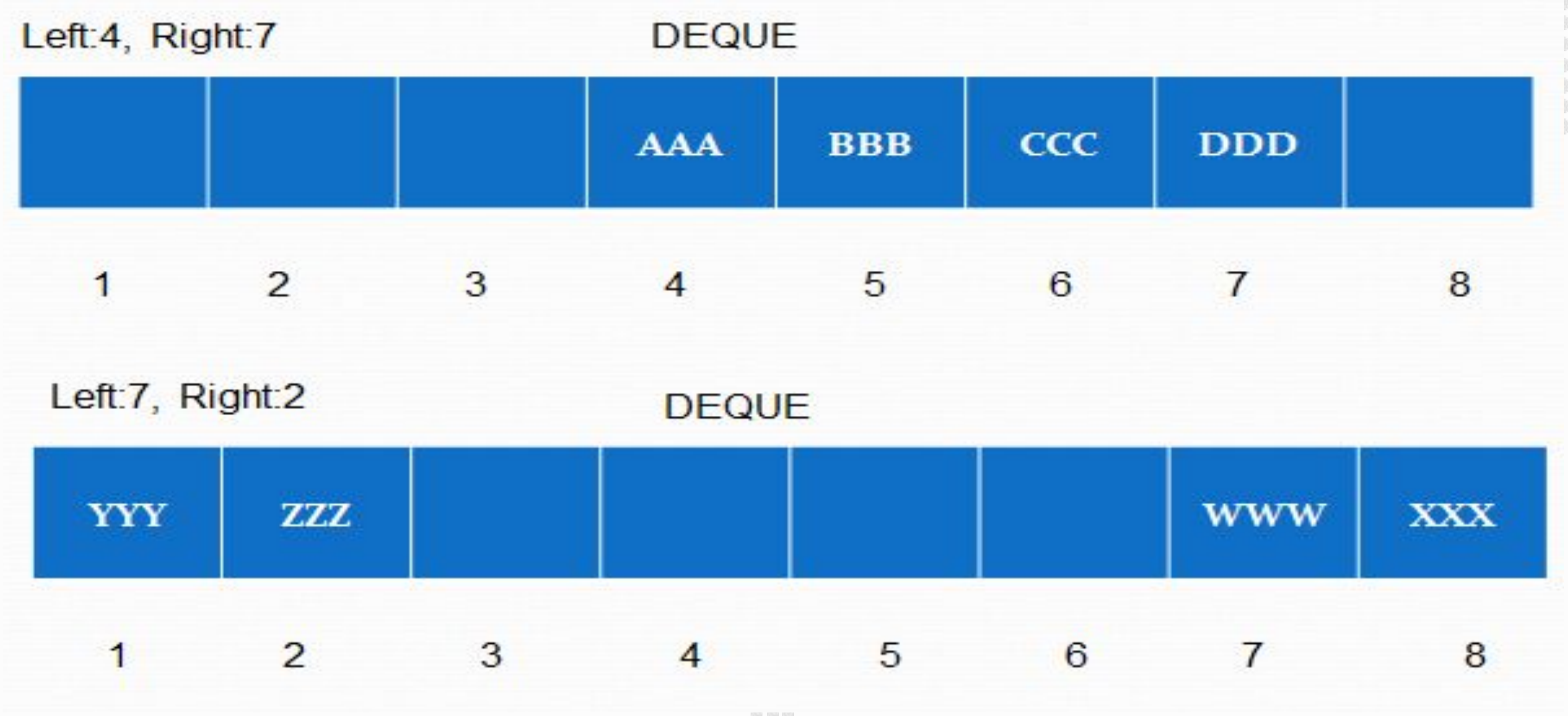
```
    QDELETE(QUEUE,N,FRONT,REAR,ITEM)
    [Queue already empty]
If FRONT=NULL, then
        Write: Underflow, and Return
Set ITEM=QUEUE[FRONT]
[Find new value of FRONT]
    If FRONT=REAR, then [Queue  has only one element to start]
        Set FRONT=NULL and REAR=NULL
    Else if FRONT =N then
        Set FRONT =1
    Else
        Set FRONT = FRONT +1
[End of if structure]
Return
```

Question 1

- Consider a circular queue with six memory cells.
- QUEUE: _,A,C,D,_,_ Front=2,REAR=4
- Add F
- Two letters are delete
- Add K,L and M
- Two letters are delete
- Add R
- Two letters are delete
- Add S
- Two letters are delete
- One letters are delete
- One letters are delete

Double Ended Queue (DEQueue)

- A DEque is a double-ended queue
- Insertions and deletions can occur at either end but not in the middle
- Implementation is similar to that for queue
- DEque are not heavily used



Double Ended Queue

There are two variations of DEque :

- **Input-restricted deque:** An input restricted deque allows insertion at only one end of the list but allows deletion a both end of the list.

- **Output-restricted deque:** An output restricted deque allows deletion at only one end of the list but allows insert in a both end of the list.

Question

Consider a circular DEQUE with six memory cells.
DEQUE:_,A,C,D,_,_ LEFT=2, RIGHT=4

a) Add F to the right
b) Two letters are deleted from the right
c) K,L, and M are added to the left
d) One letter on the left is deleted
e) R is added to the left
f) S is added to the right
g) T is added to the right

Priority DEqueue

- A priority queue is a collection of elements such that each element has been assigned a priority and such that the order in which the elements are deleted and processed comes from the following rules:
- An element of higher priority is processed before any element of lower priority.
- Two elements of the same priority are processed according to the order in which they were added to the queue.

Priority Queue

- Sometimes, we have queues that are not FIFO i.e. the person who comes first may not leave first.
- We can develop such queues in which the condition for leaving the queue is not to enter first. There may be some priority.
- We will develop the queue in such a way that we will get the event which is going to happen first of all in the future. This data structure is known as priority queue.
- In a sense, FIFO is a special case of priority queue in which priority is given to the time of arrival.
- In operating systems, we have queue of different processes. If some process comes with higher priority, it will be processed first.

Polish Notation

- Polish notation named after Polish mathematician Jan Lukasiewiez, refers to the notation in which the operator symbol is placed before its operands.

- Reverse Polish notation refers to the notation in which operator symbol is placed after its operands.

Evaluation of Expression

An expression is made up of operands, operators, and delimiters.

A/B–C+D*E–A*C

Arithmetic operators

+,–,*,/, %, and unary minus

Relational operators

<,<=,==,<.,>=, >, &&, ||, and !.

Priority of Operators

To fix the order of evaluation, each operator is assigned a priority.

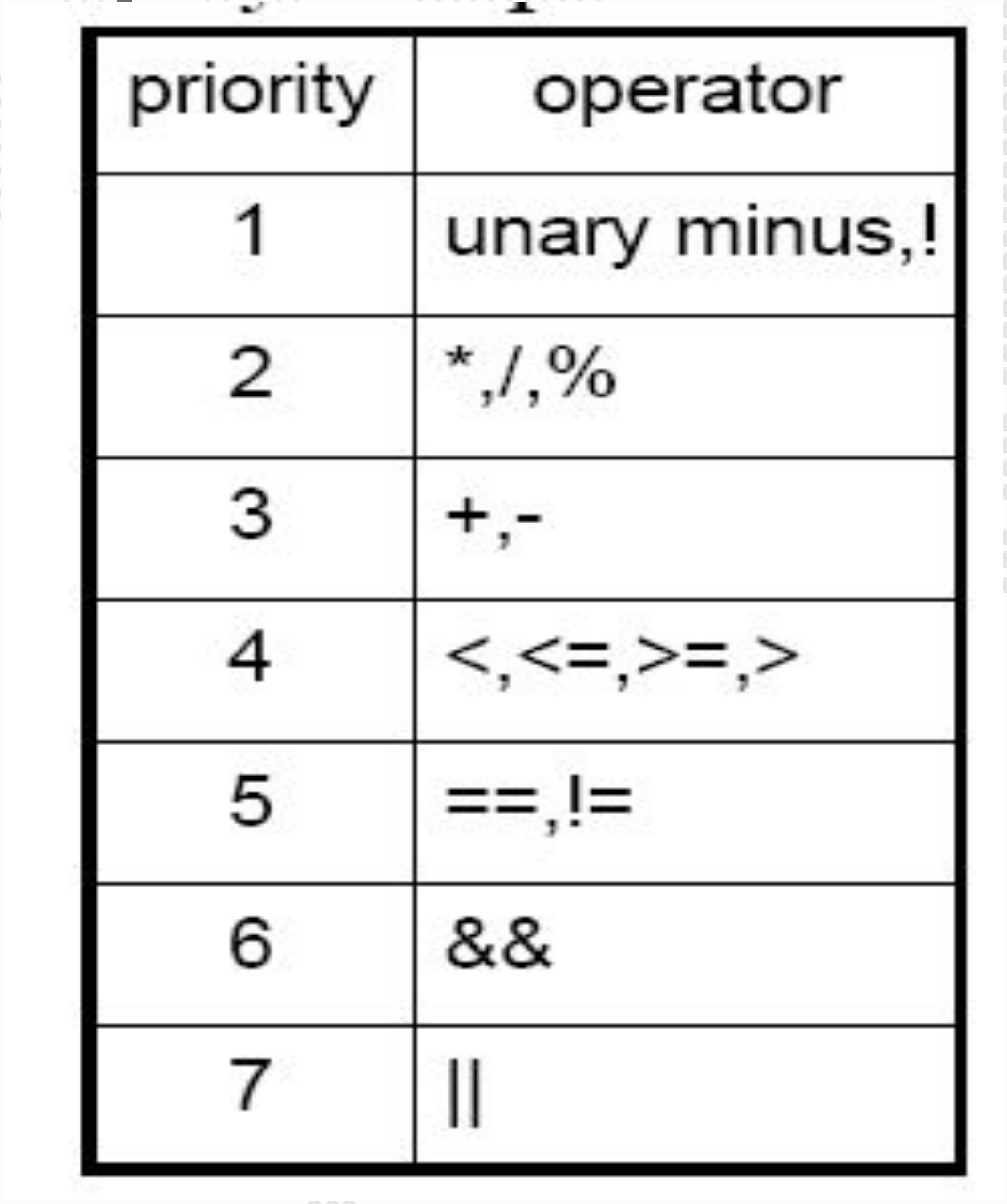
The C++ rule is that for all priorities, evaluation of operators of the same priority will proceed left to right.

A/B*C will be evaluated as (A/B)*C.

X=A/B–C+D*E–A*C will be evaluated as

X=(((A/B)–C)+(D*E))–(A*C).

Priority of Operators



priority	operator
1	unary minus,!
2	*,/,%
3	+,-
4	<,<=,>=,>
5	==,!=
6	&&
7	||

Postfix Notation

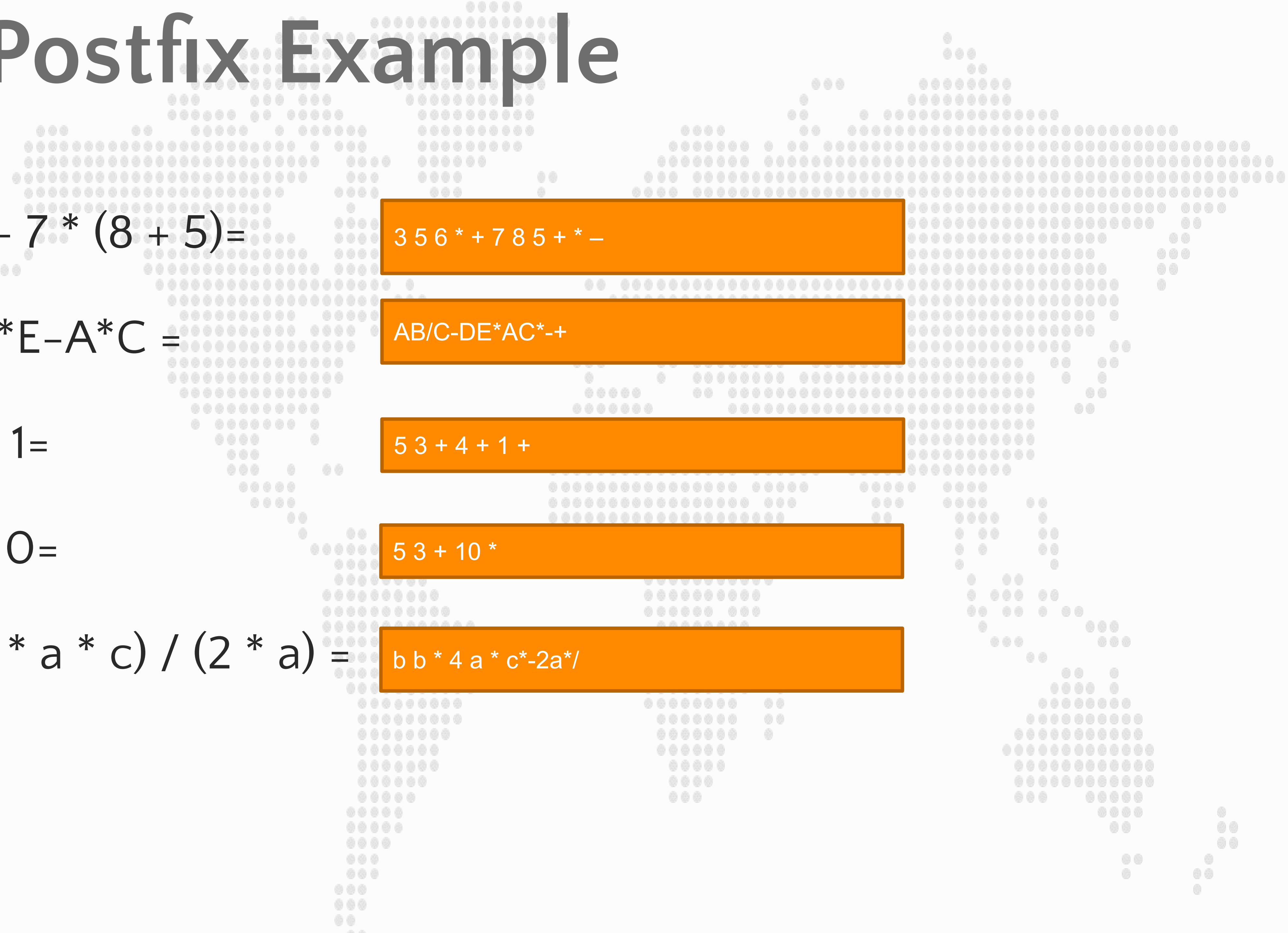
A compiler accepts an expression and produces correct code by reworking the expression into a form called postfix notation. The conventional way of writing an expression is called infix-the operators come in-between the operands

Infix A*B/C has postfix AB*C/.

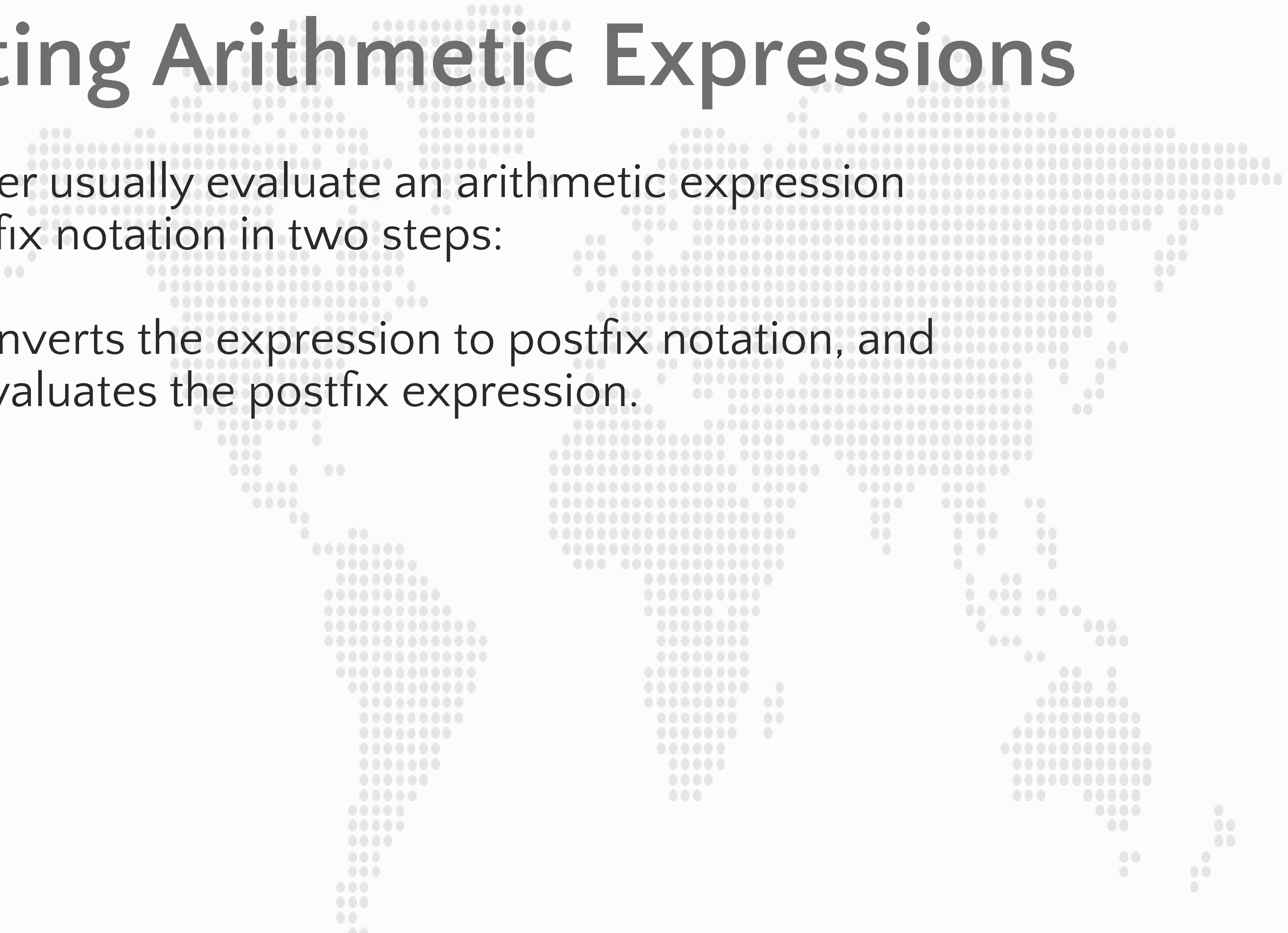
Infix: A/B-C+D*E-A*C

Postfix: AB/C-DE*+AC*-

Infix – Postfix Example



• 3 + 5 * 6 – 7 * (8 + 5)=	3 5 6 * + 7 8 5 + * –
• A/B–C+D*E–A*C =	AB/C-DE*AC*-+
• 5+ 3 + 4 + 1=	5 3 + 4 + 1 +
• (5 + 3) * 10=	5 3 + 10 *
• (b * b – 4 * a * c) / (2 * a) =	b b * 4 a * c*-2a*/



Evaluating Arithmetic Expressions

The computer usually evaluate an arithmetic expression written in infix notation in two steps:

- ❑ First it converts the expression to postfix notation, and
- ❑ Then it evaluates the postfix expression.