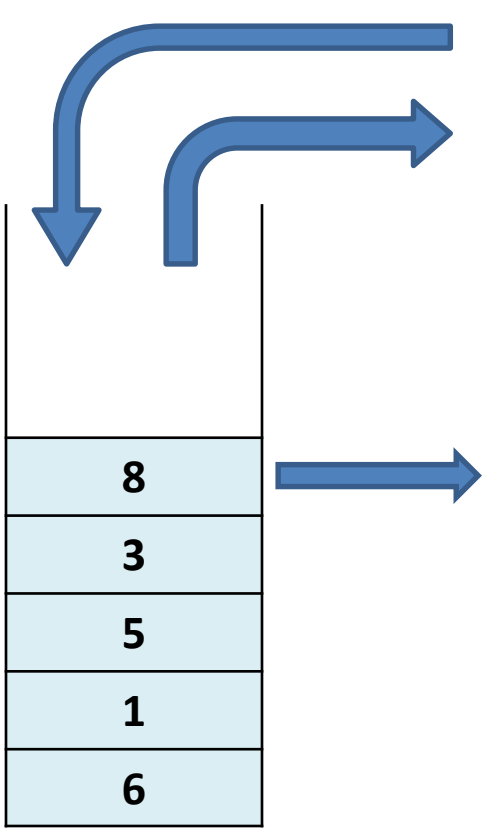
Containers: Stack

Jordi Cortadella and Jordi Petit
Department of Computer Science

The Stack ADT

- A stack is a list of objects in which insertions and deletions can only be performed at the top of the list.
- Also known as LIFO (Last In, First Out)



push (insert an element at the top)

pop (delete the most recently inserted element)

top (the most recently inserted element)

empty (is there any element?)

Note: pop and top generate an error on an empty stack

The Stack ADT

```
template <typename T>
class Stack {
  public:
    // Default constructor
    Stack() {}
    …
  private:
    vector<T> data;
};
```

- The definition can handle generic stacks of any type T.
- The default constructor does not need to do anything:
  a zero-sized vector is constructed by default.

The Stack ADT

```
template <typename T>
class Stack {
  public:
    …
    bool empty() const {
      return data.size() == 0;
    }

    const T& top() const { // Returns a const reference
      assert (not empty());
      return data.back();
    }

    T& top() {             // Returns a reference
      assert (not empty());
      return data.back();
    }

    void pop() {
      assert (not empty());
      data.pop_back();
    }

    void push(const T& x) {
      data.push_back(x);
    }
};
```

Balancing symbols

- **Balancing symbols**: check for syntax errors when expressions have opening/closing symbols, e.g., () [] {}

  Correct: [(){()[()]}()]
  Incorrect: [(){(}…

- **Algorithm** (linear): read all chars until end of file. For each char, do the following:
  - If the char is opening, push it onto the stack.
  - If the char is closing and stack is empty → error, otherwise pop a symbol from the stack and check they match. If not → error.
  - At the end of the file, check the stack is empty.

- **Exercise:** implement and try the above examples.

Evaluation of postfix expressions

- This is an infix expression. What's his value? 42 or 968?

  $8 * 3 + 10 + 2 * 4$

- It depends on the operator precedence. For scientific calculators, * has precedence over +.

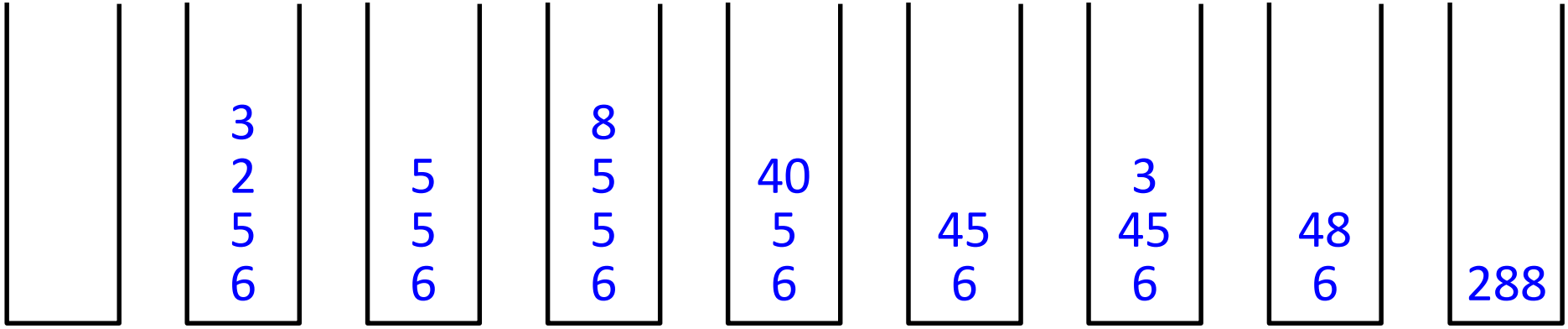
- Postfix (reverse Polish notation) has no ambiguity:

  $8\ 3 *\ 10\ +\ 2\ 4 *\ +$

- Postfix expressions can be evaluated using a stack:
  - each time an operand is read, it is pushed on the stack
  - each time an operator is read, the two top values are popped and operated. The result is push onto the stack

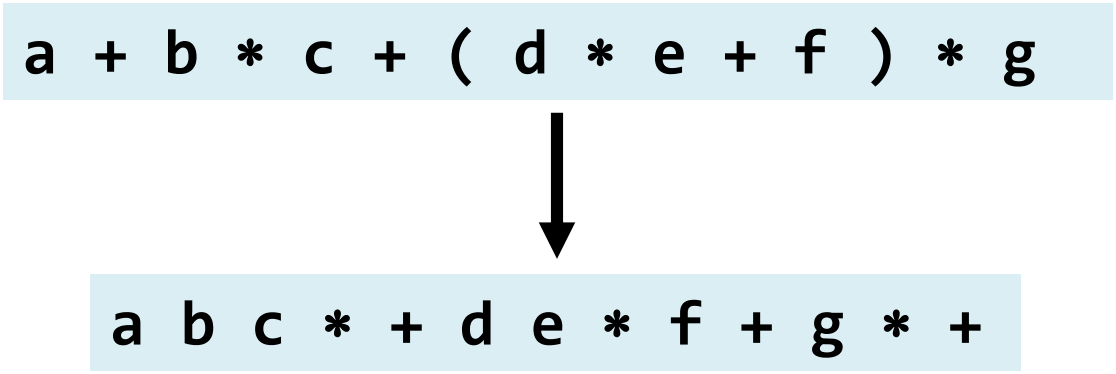
Evaluation of postfix expressions: example

6 5 2 3 + 8 * + 3 + *



push(6)
push(5)
push(2)
push(3)
+
push(8)
*
+
push(3)
+
*

From infix to postfix



Algorithm:

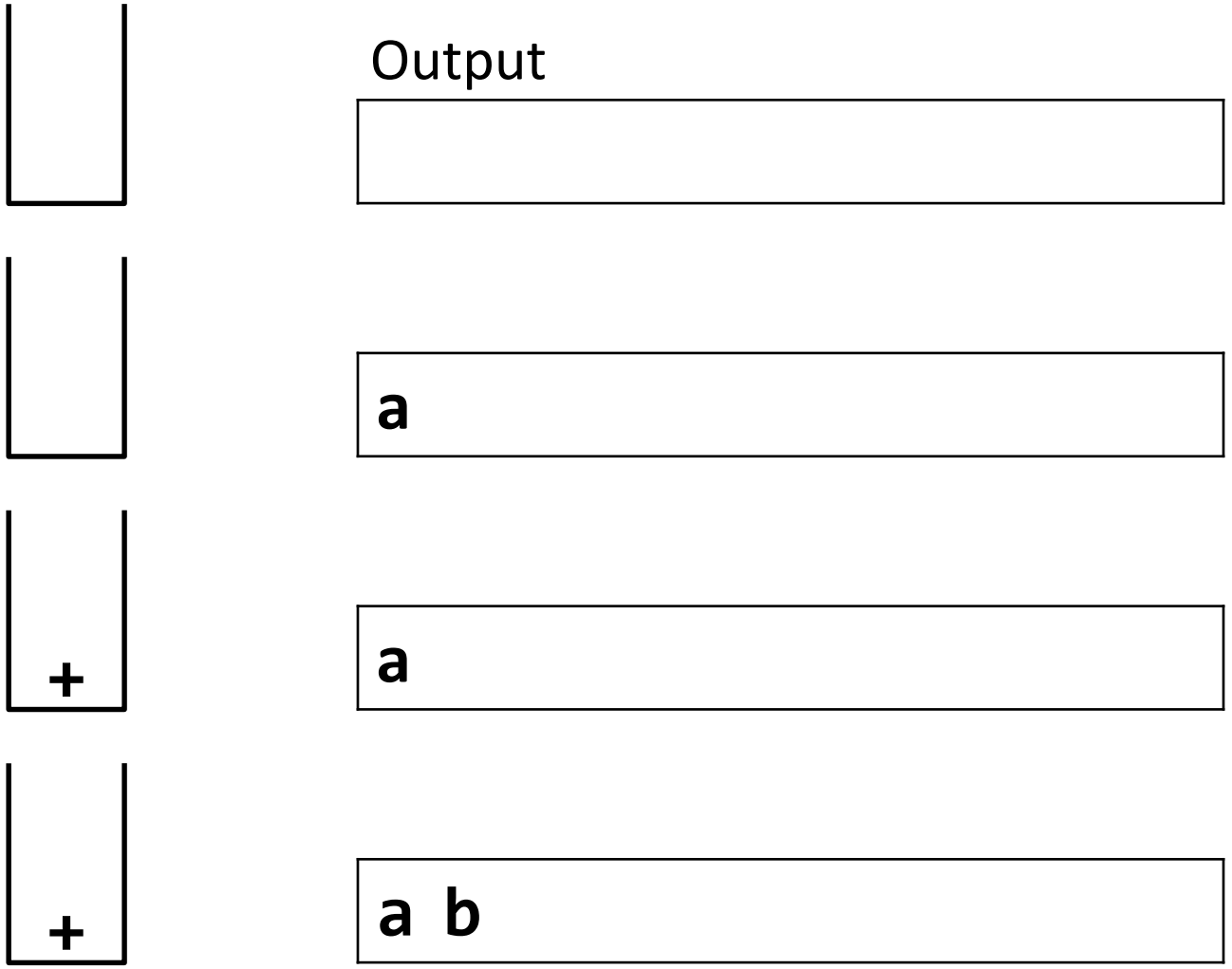
- When an operand is read, write it to the output.
- If we read a right parenthesis, pop the stack writing symbols until we encounter the left parenthesis.
- For any other symbol (‘+’, ‘*’, ‘(‘), pop entries and write them until we find an entry with lower priority. After popping, push the symbol onto the stack. Exception: ‘(‘ can only be removed when finding a ‘)’.
- When the end of the input is reached, all symbols in the stack are popped and written onto the output.

From infix to postfix

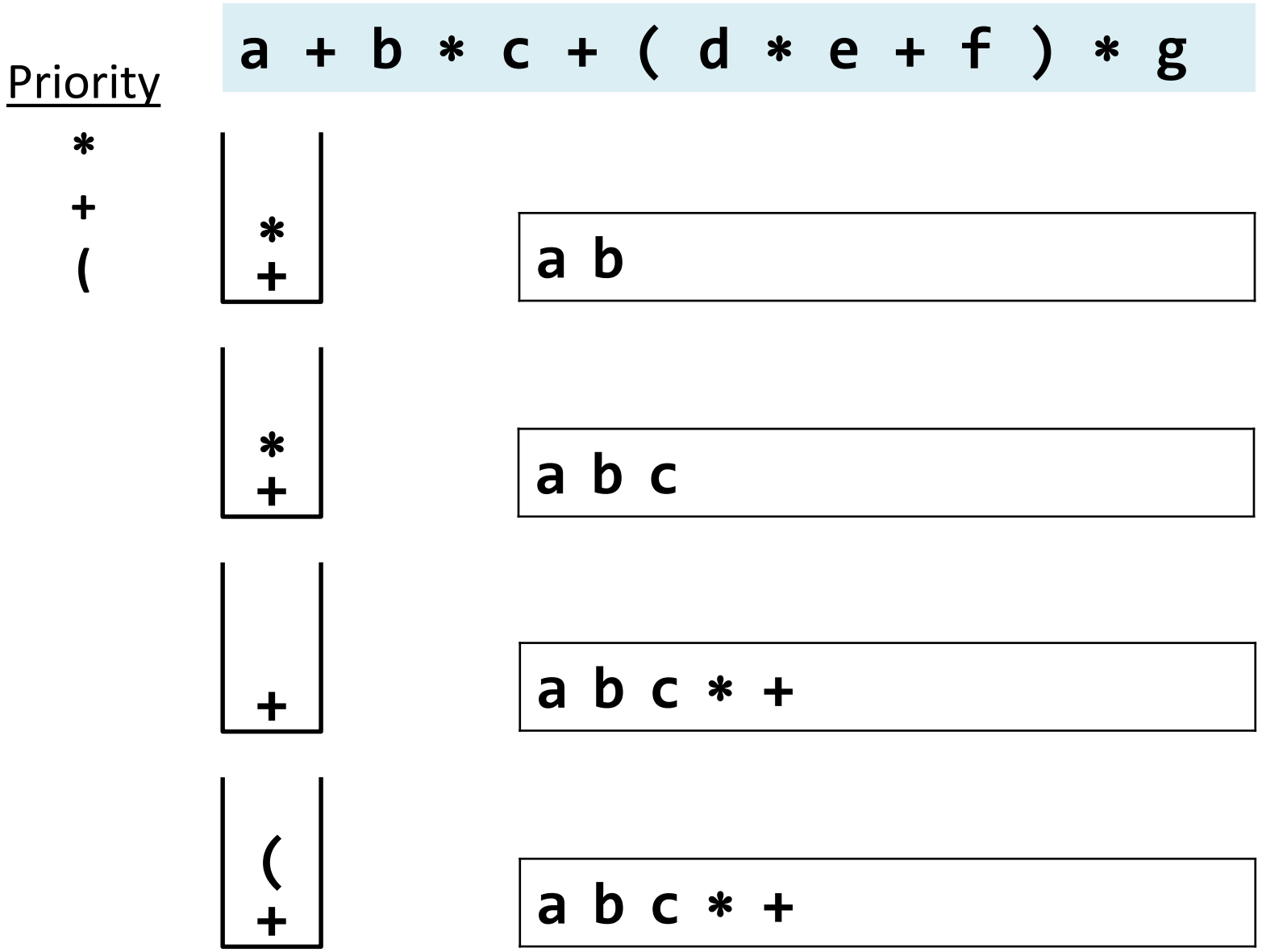
a + b * c + (d * e + f) * g

Priority

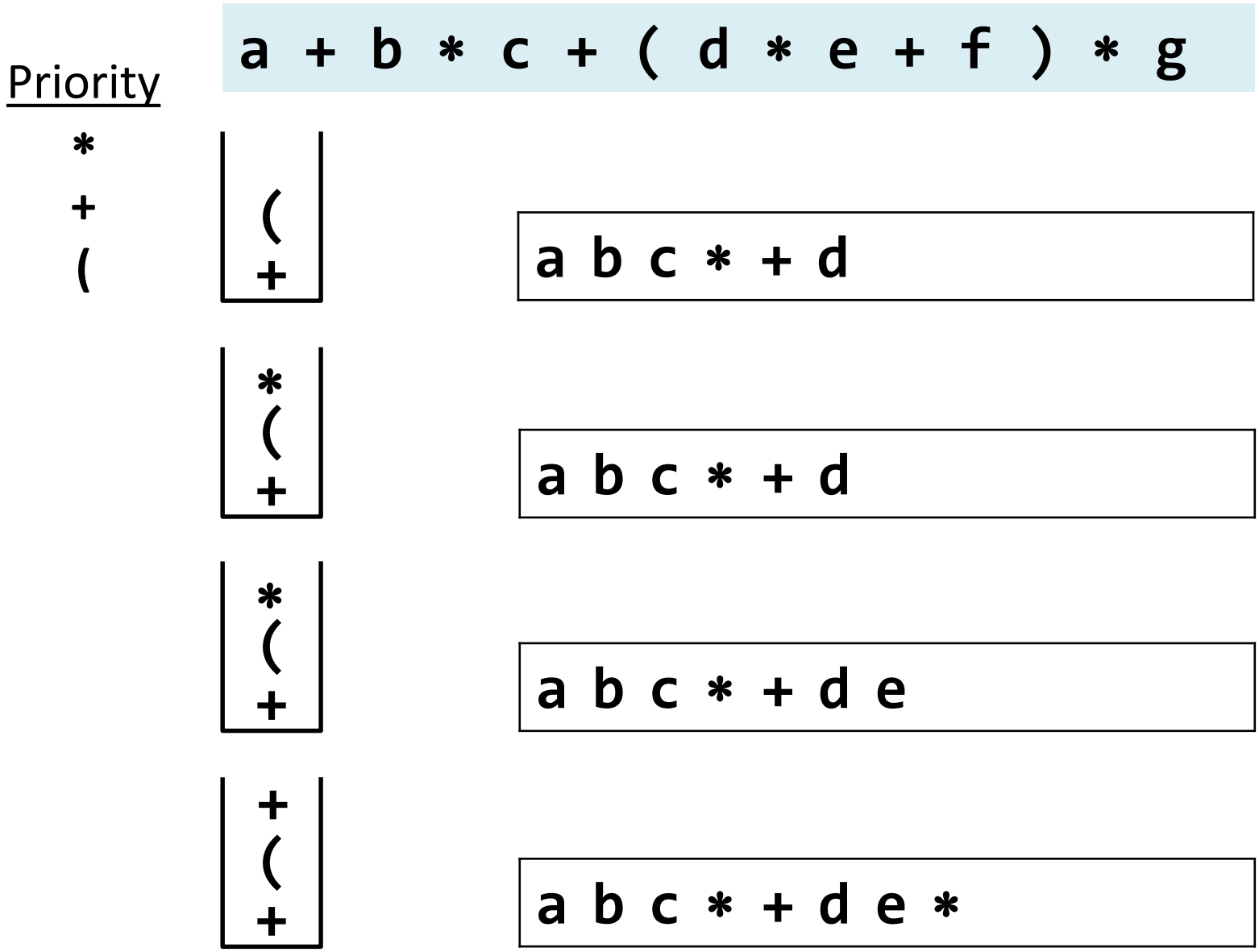
*
+
(



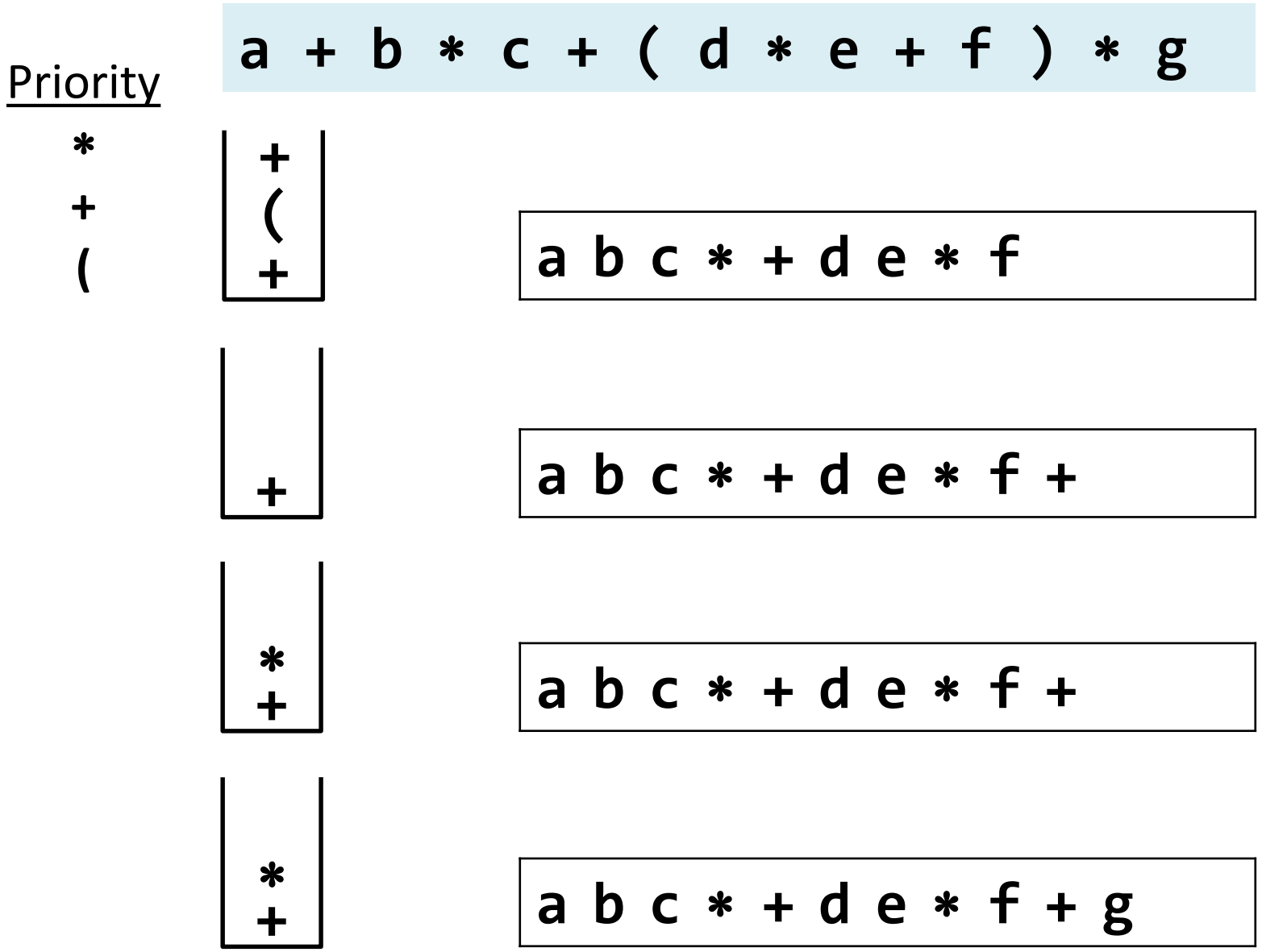
From infix to postfix



From infix to postfix



From infix to postfix



From infix to postfix

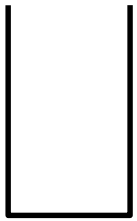
a + b * c + (d * e + f) * g

Priority

*

+

(



a b c * + d e * f + g * +

Complexity: $O(n)$

Suggested exercise:

- Add substraction (same priority as addition) and division (same priority as multiplication).

EXERCISES

Interleaved push/pop operations

Suppose that an intermixed sequence of push and pop operations are performed. The pushes push the integers 0 through 9 in order; the pops print out the return value. Which of the following sequences could not occur?

a) 4 3 2 1 0 9 8 7 6 5
b) 4 6 8 7 5 3 2 9 0 1
c) 2 5 6 7 4 8 9 3 1 0
d) 4 3 2 1 0 5 6 7 8 9

Source: Robert Sedgewick, Computer Science 126, Princeton University.

Middle element of a stack

Design the class **`MidStack`** implementing a stack with the following operations:

- Push/pop: the usual operations on a stack.
- FindMiddle: returns the value of the element in the middle.
- DeleteMiddle: deletes the element in the middle.

All the operations must be executed in $O(1)$ time.

Suggestion: use some container of the STL to implement it.

Note: if the stack has n elements at locations $0..n-1$, where 0 is the location at the bottom, the middle element is the one at location $\lfloor (n-1)/2 \rfloor$.