http://algs4.cs.princeton.edu

1.3 DIJKSTRA'S 2-STACK DEMO

- *With correction*

Dijkstra's two-stack algorithm

Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore (but see how we use them in the repo).

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.

value stack

operator stack

infix expression

(fully parenthesized)

Dijkstra's two-stack algorithm

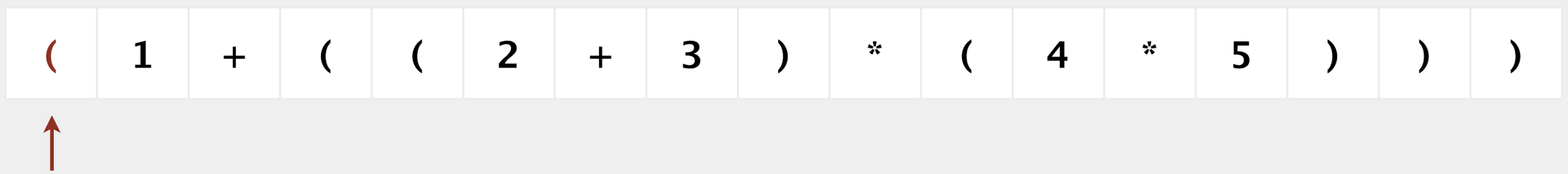
Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.

value stack operator stack



Dijkstra's two-stack algorithm

Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.

value stack operator stack

(	1	+	(	(	2	+	3	)	*	(	4	*	5	)	)	)

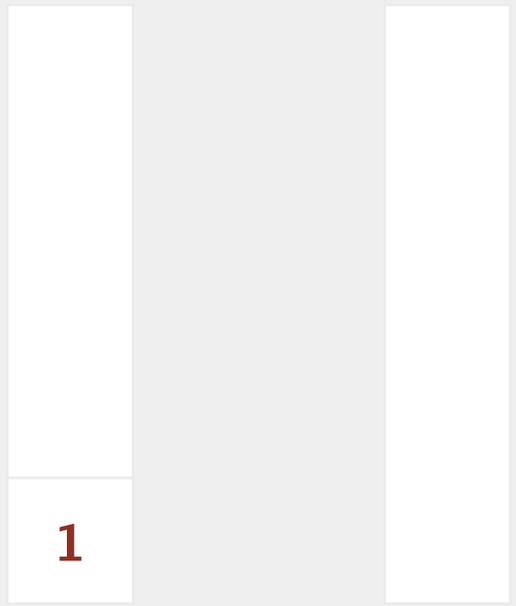
Dijkstra's two-stack algorithm

Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.



(	1	+	(	(	2	+	3	)	*	(	4	*	5	)	)	)

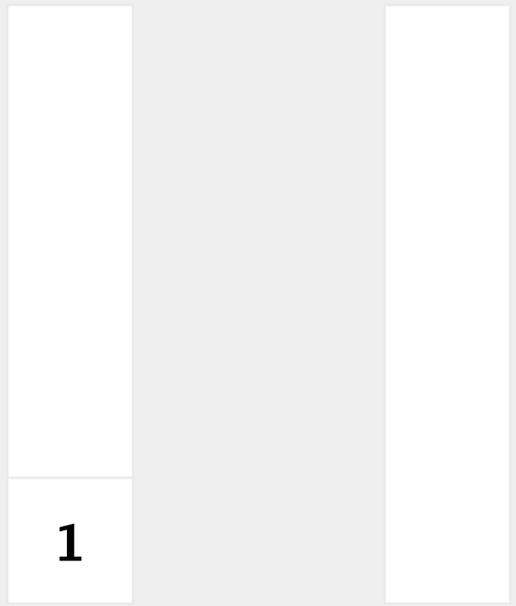
Dijkstra's two-stack algorithm

Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.



value stack operator stack

(	1	+	(	(	2	+	3	)	*	(	4	*	5	)	)	)

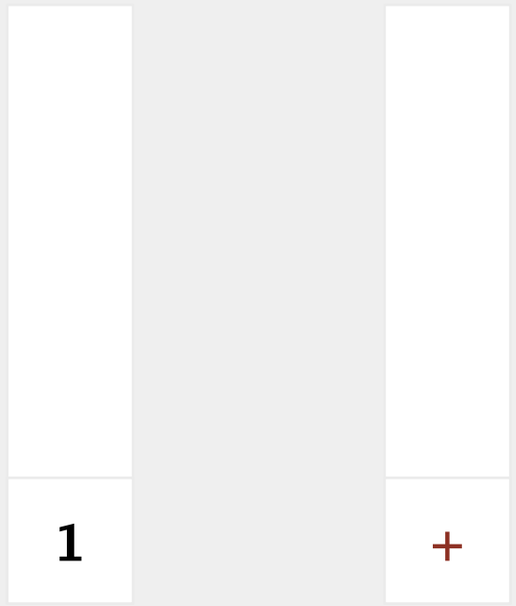
Dijkstra's two-stack algorithm

Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.



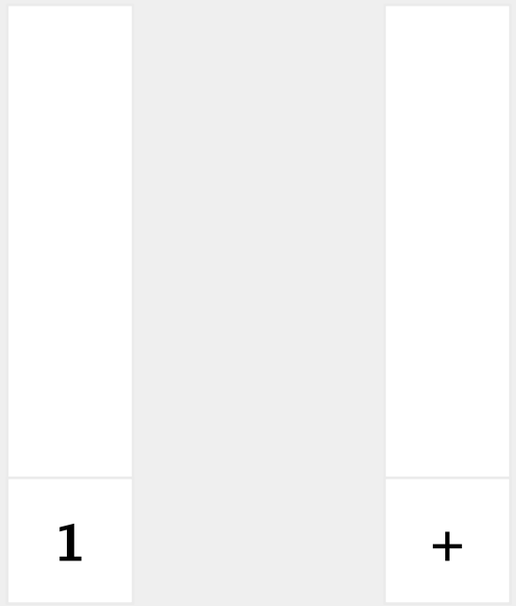
Dijkstra's two-stack algorithm

Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.



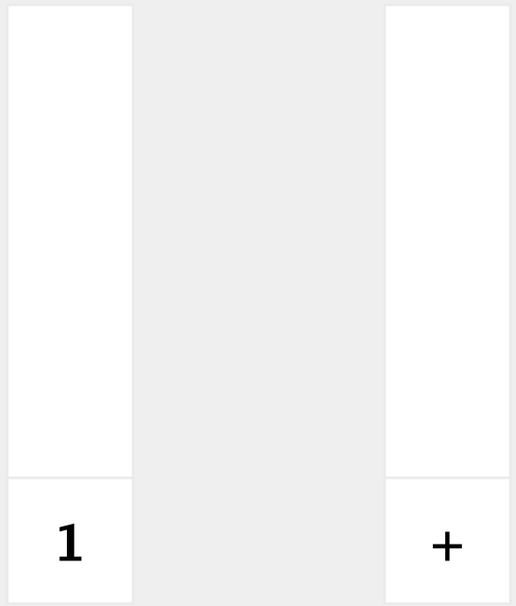
Dijkstra's two-stack algorithm

Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.



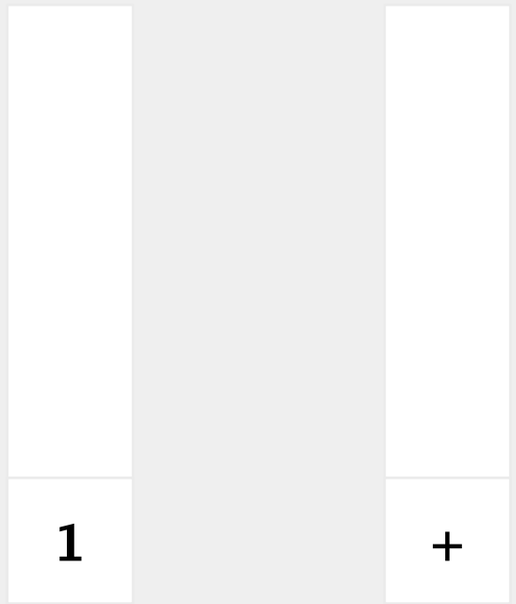
Dijkstra's two-stack algorithm

Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.



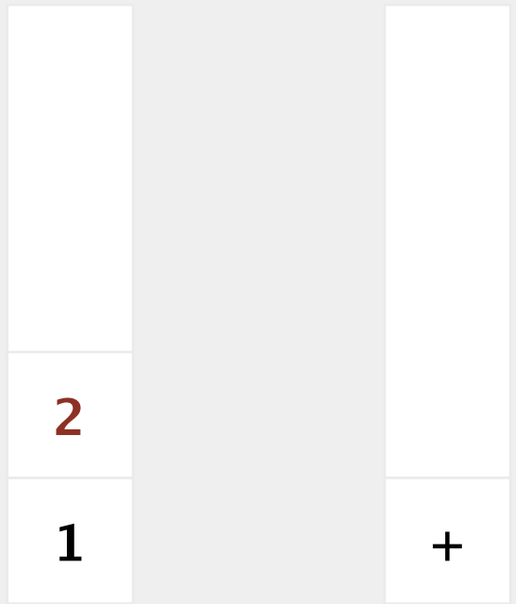
Dijkstra's two-stack algorithm

Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.



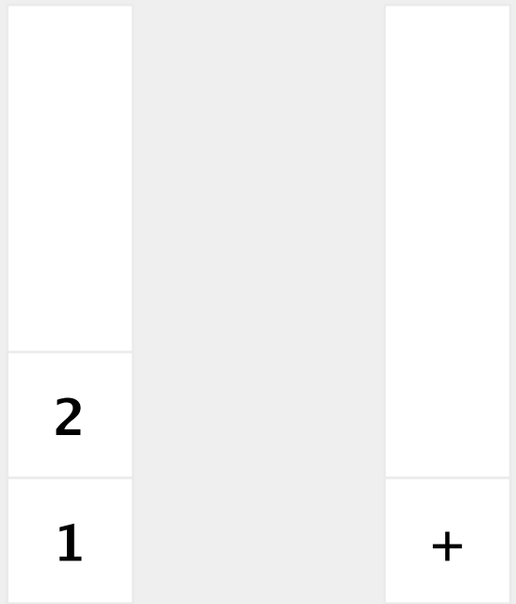
Dijkstra's two-stack algorithm

Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.



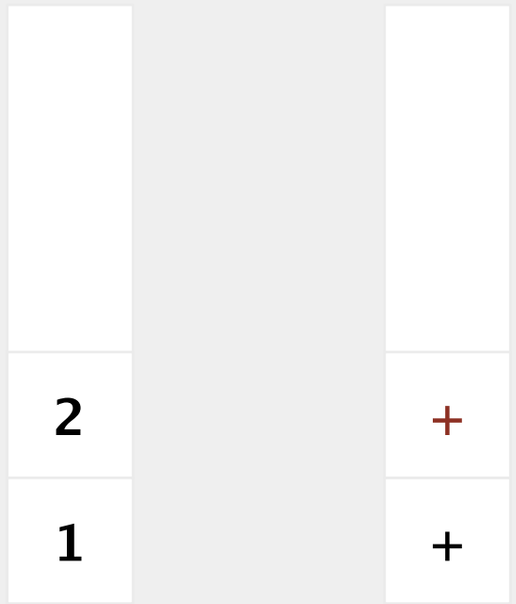
Dijkstra's two-stack algorithm

Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.



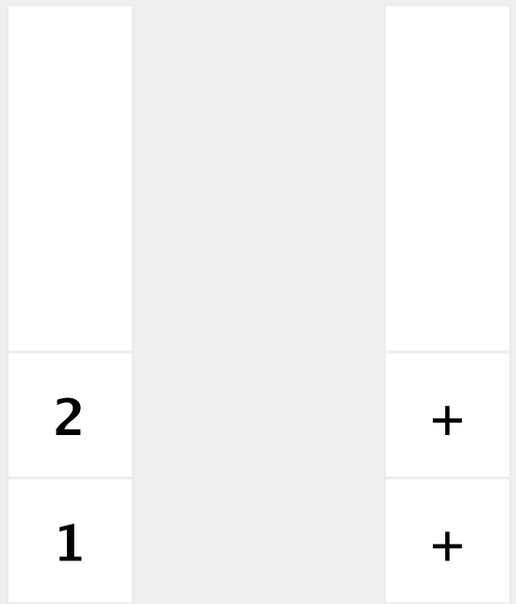
Dijkstra's two-stack algorithm

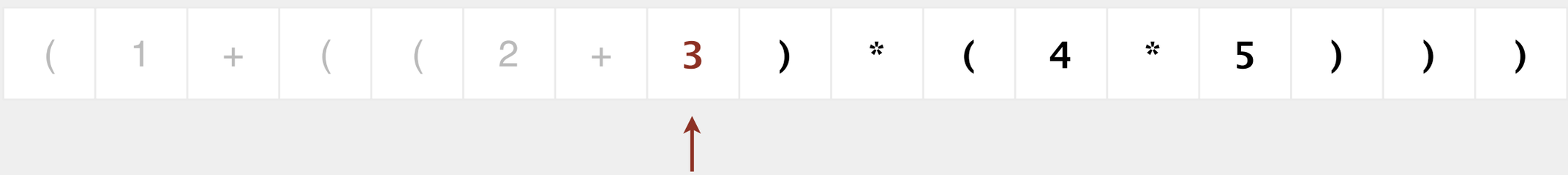
Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.





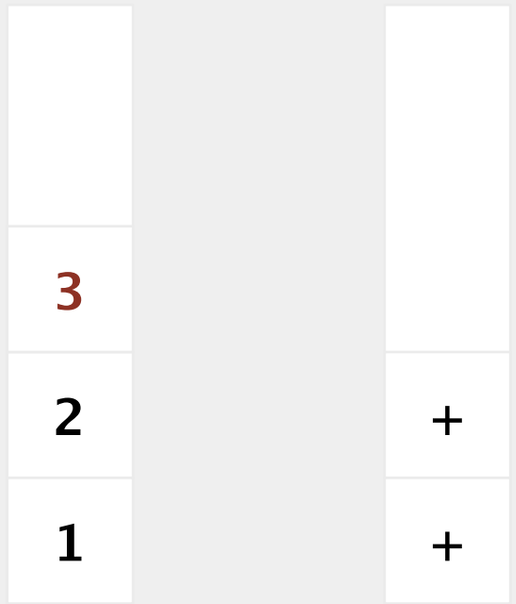
Dijkstra's two-stack algorithm

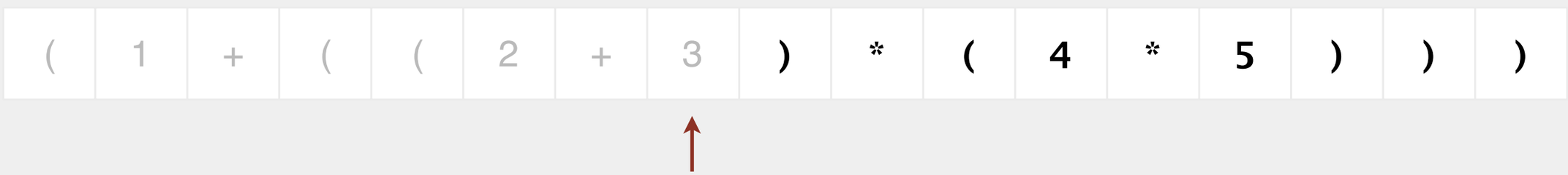
Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.





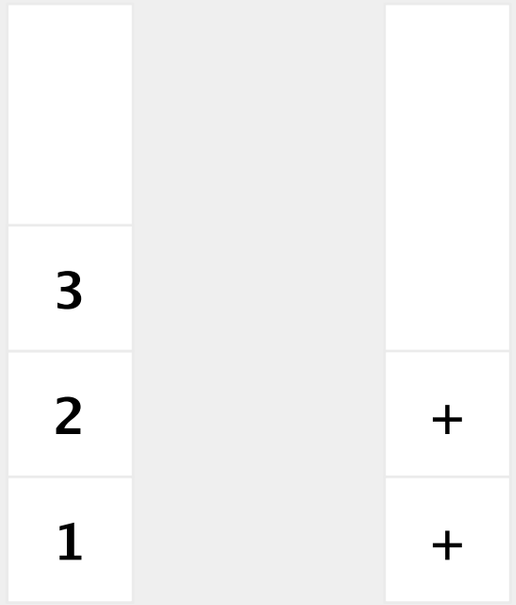
Dijkstra's two-stack algorithm

Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.



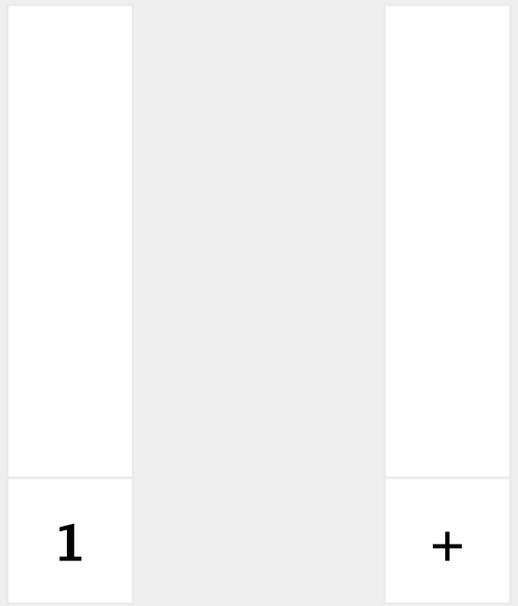
Dijkstra's two-stack algorithm

Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.



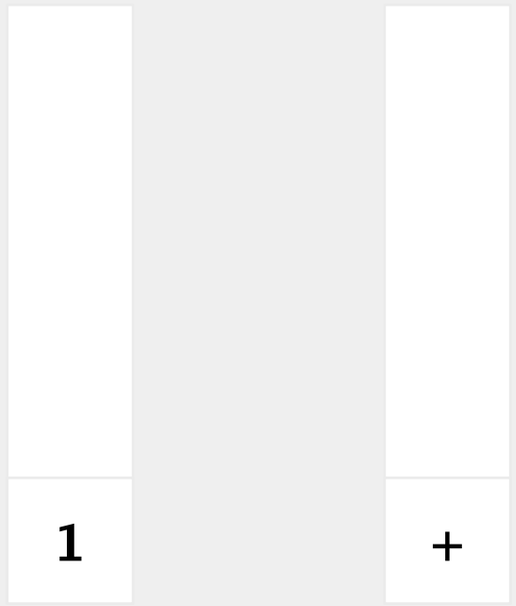
Dijkstra's two-stack algorithm

Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.



3	+	2	=	5

(	1	+	(	(	2	+	3	)	*	(	4	*	5	)	)	)

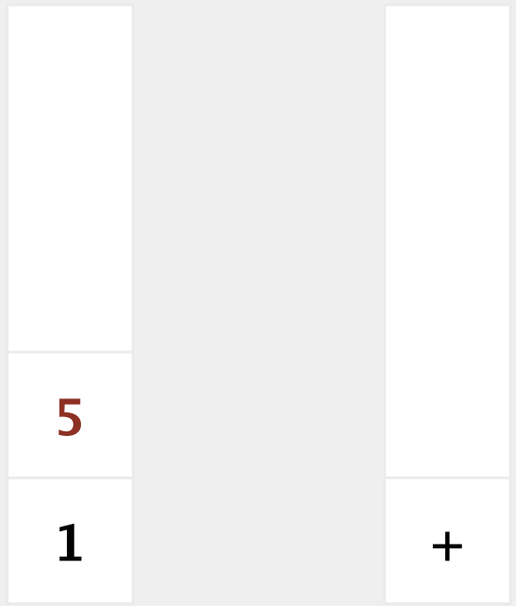
Dijkstra's two-stack algorithm

Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.



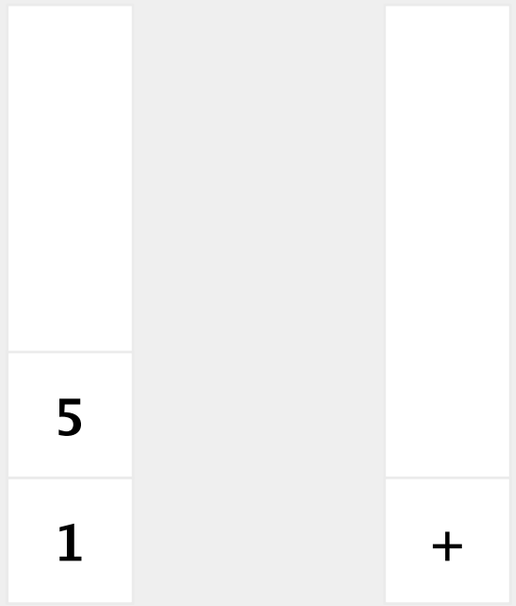
Dijkstra's two-stack algorithm

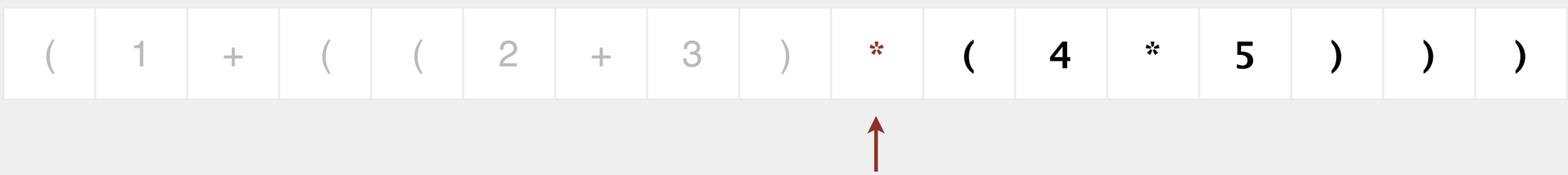
Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.





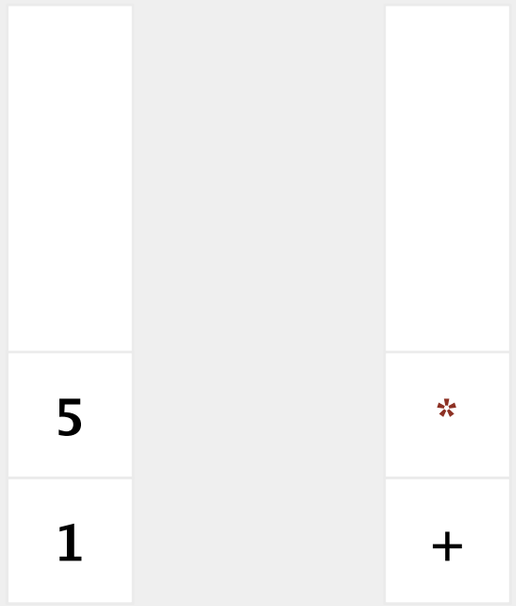
Dijkstra's two-stack algorithm

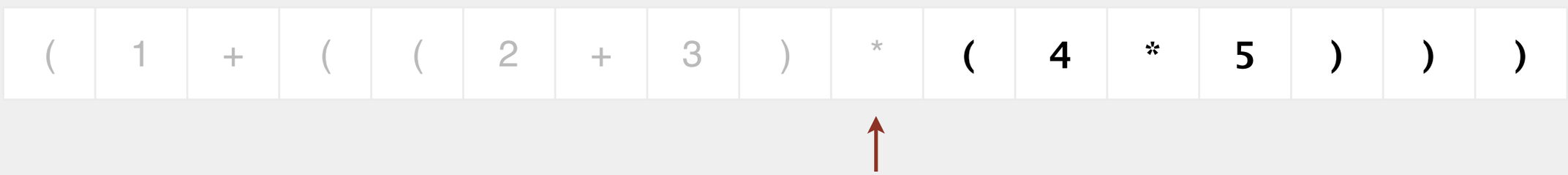
Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.





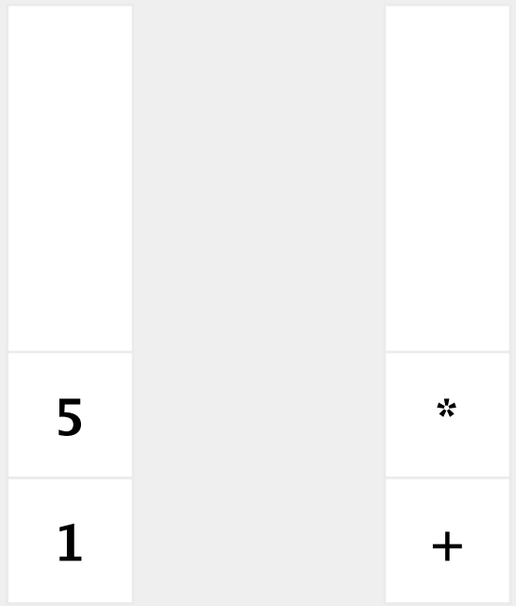
Dijkstra's two-stack algorithm

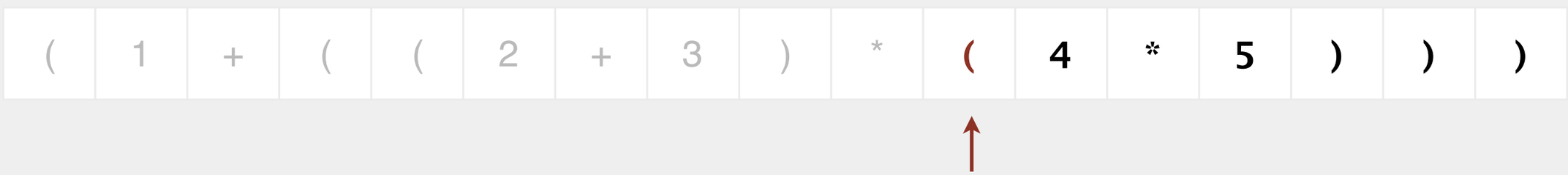
Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.





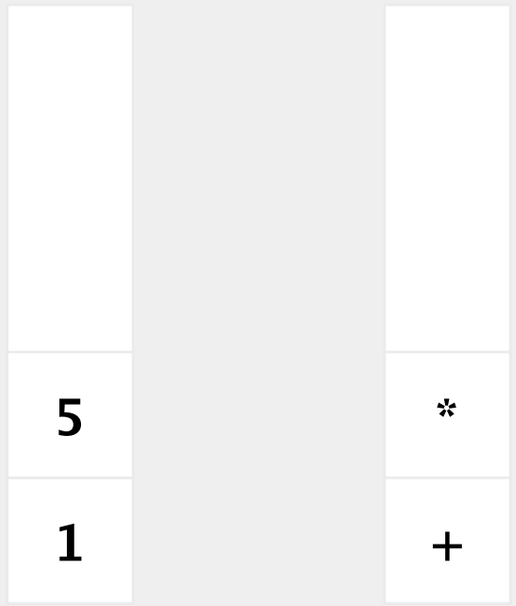
Dijkstra's two-stack algorithm

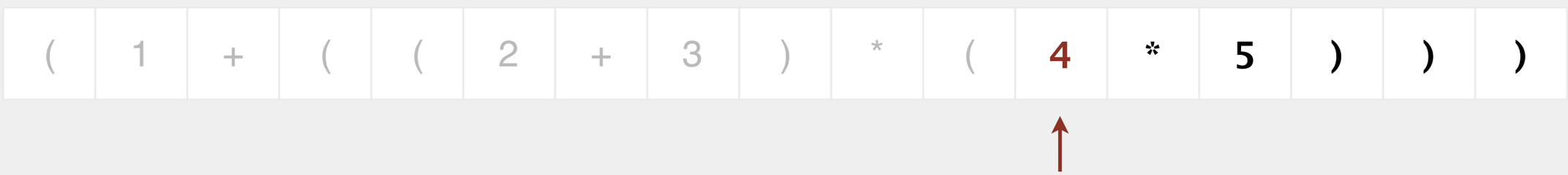
Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.





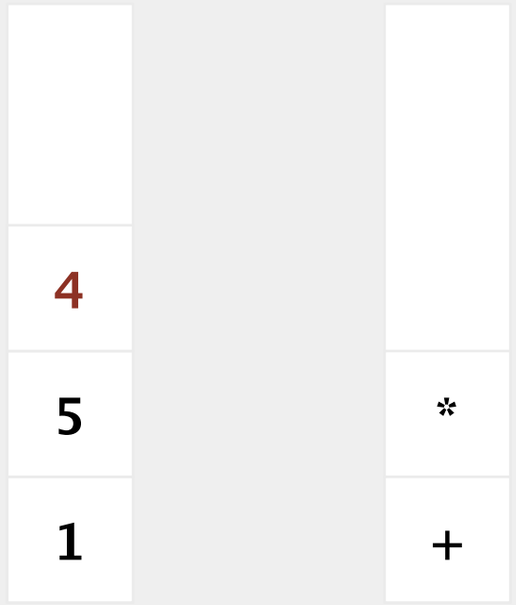
Dijkstra's two-stack algorithm

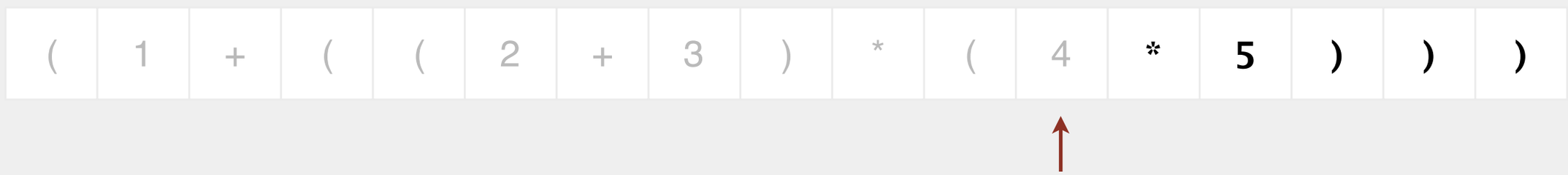
Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.





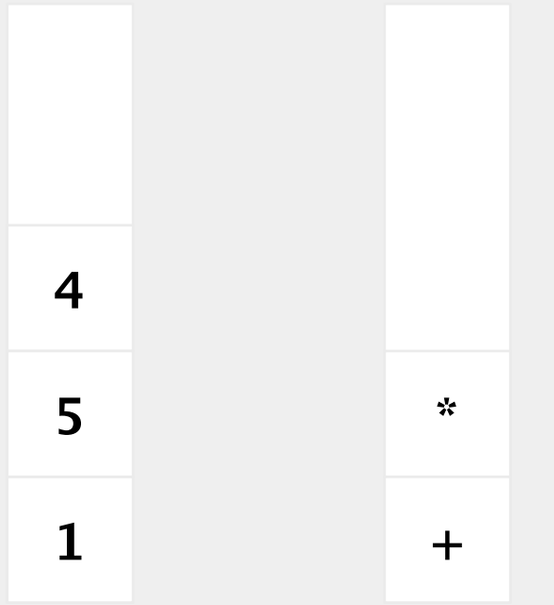
Dijkstra's two-stack algorithm

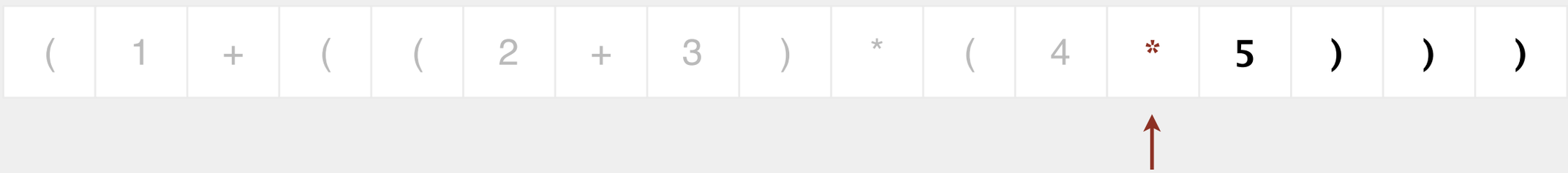
Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.





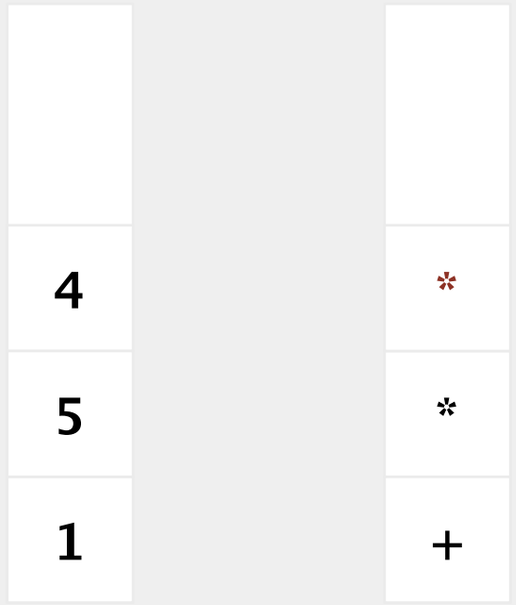
Dijkstra's two-stack algorithm

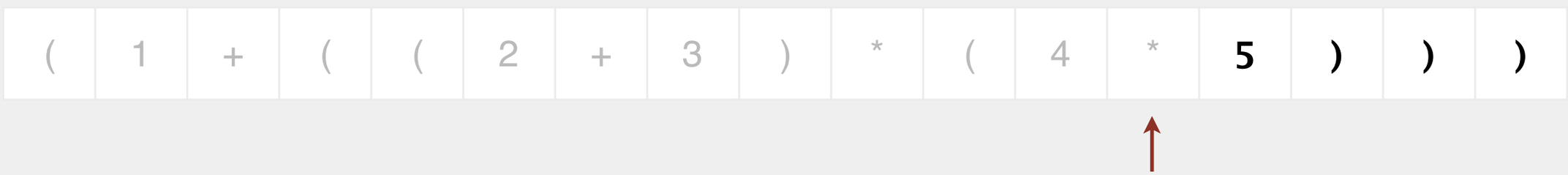
Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.





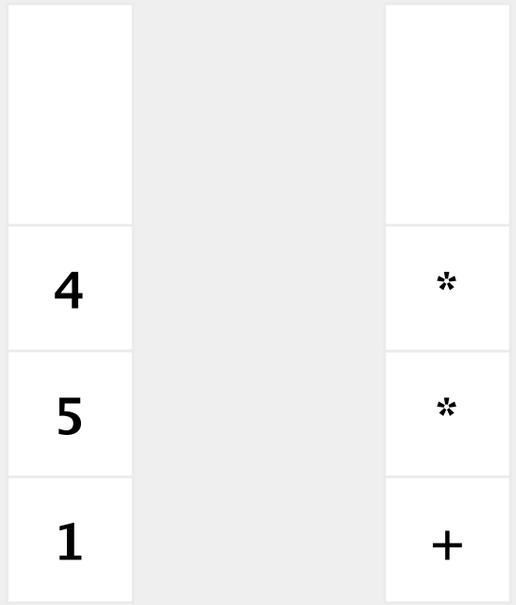
Dijkstra's two-stack algorithm

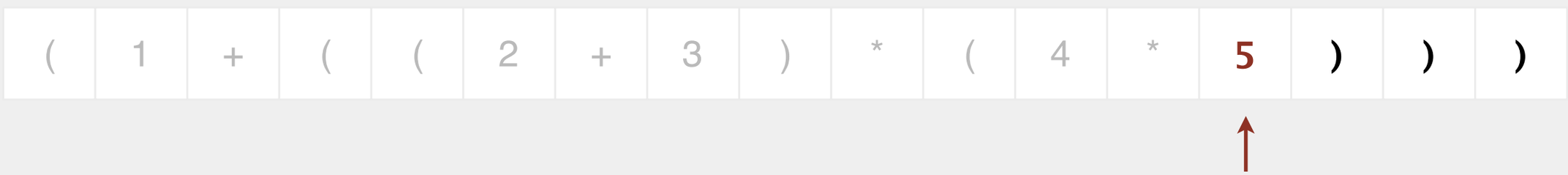
Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.





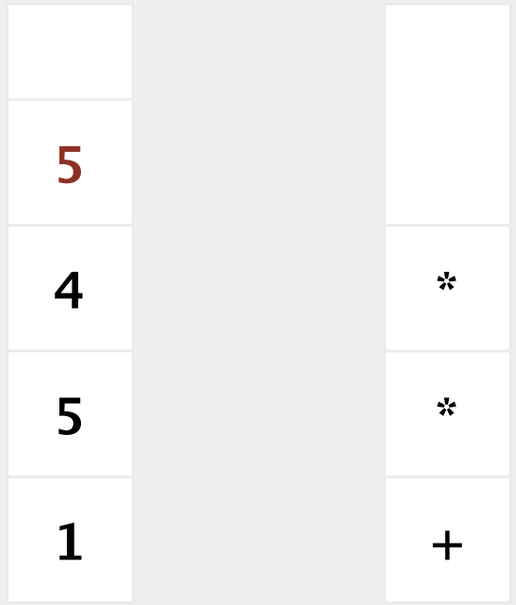
Dijkstra's two-stack algorithm

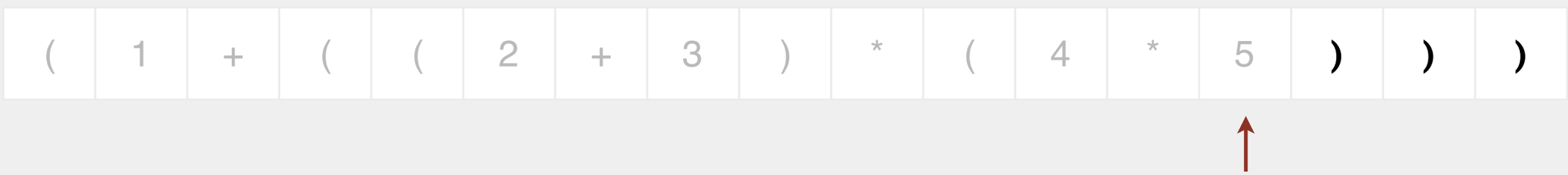
Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.





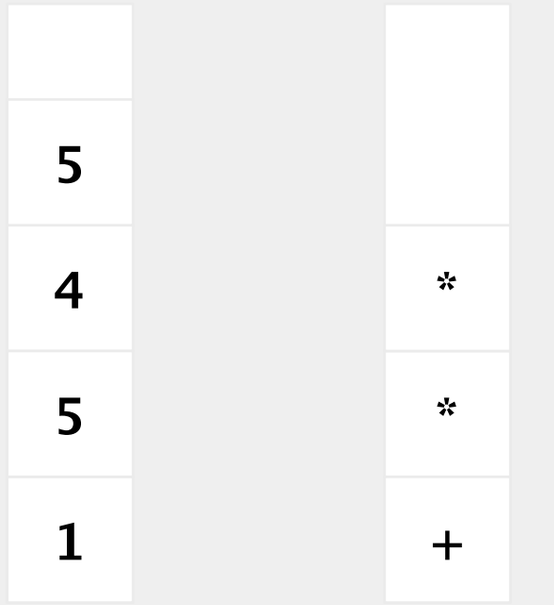
Dijkstra's two-stack algorithm

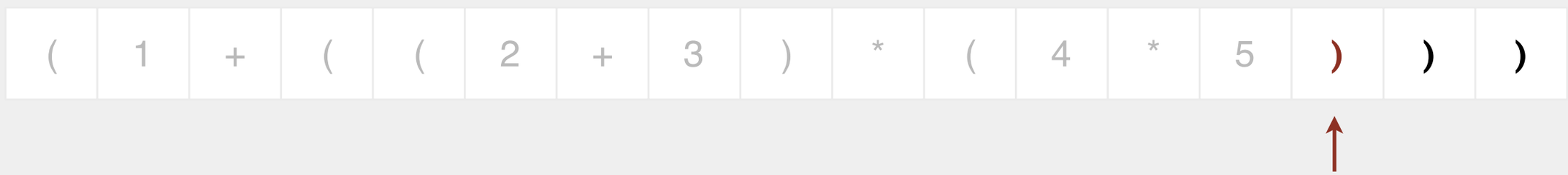
Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.





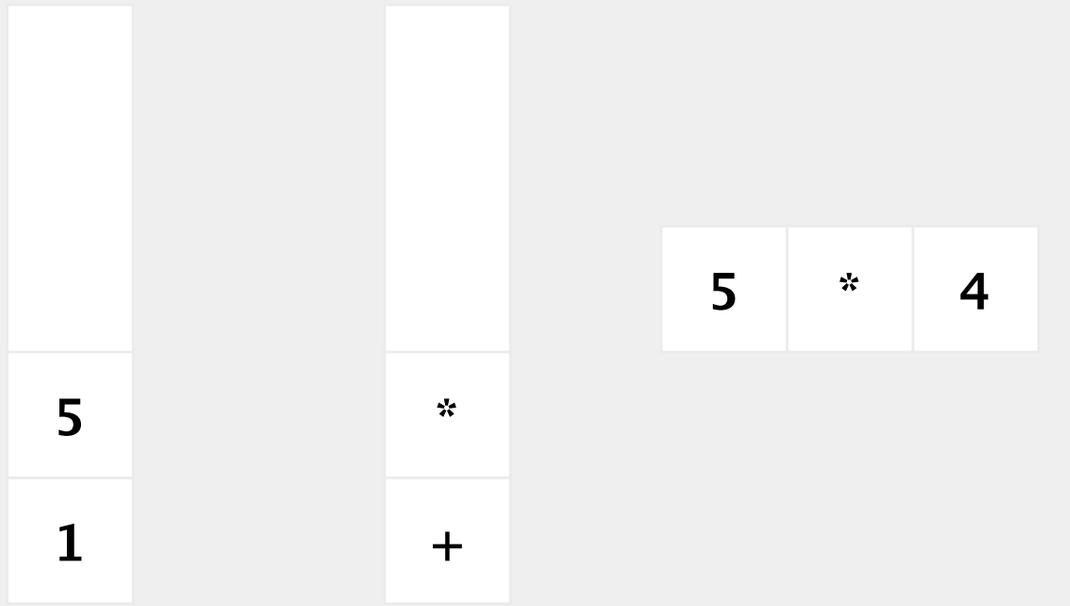
Dijkstra's two-stack algorithm

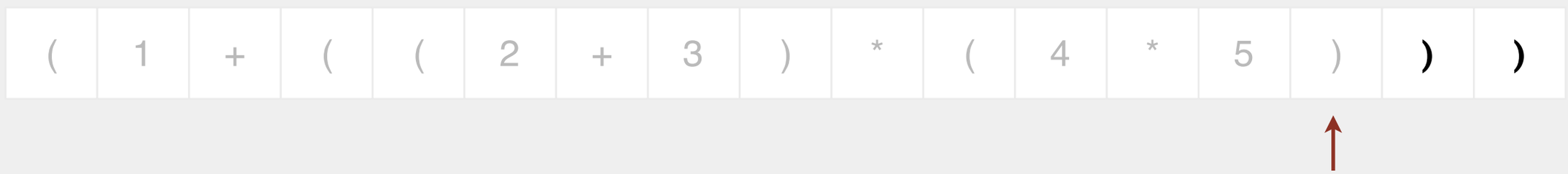
Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.





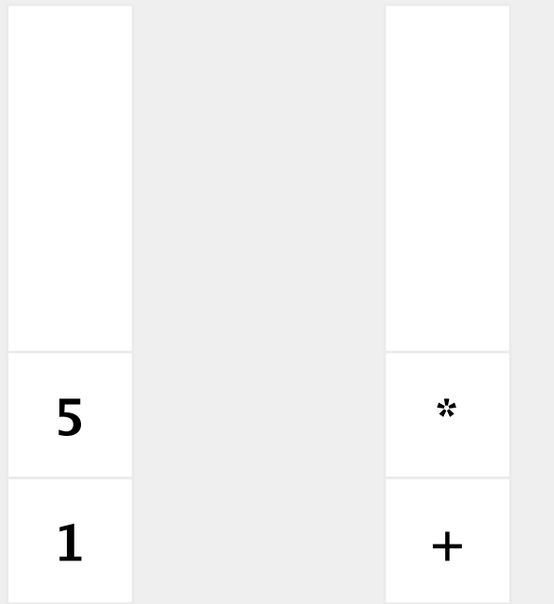
Dijkstra's two-stack algorithm

Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.



5	*	4	=	20

(	1	+	(	(	2	+	3	)	*	(	4	*	5	)	)	)

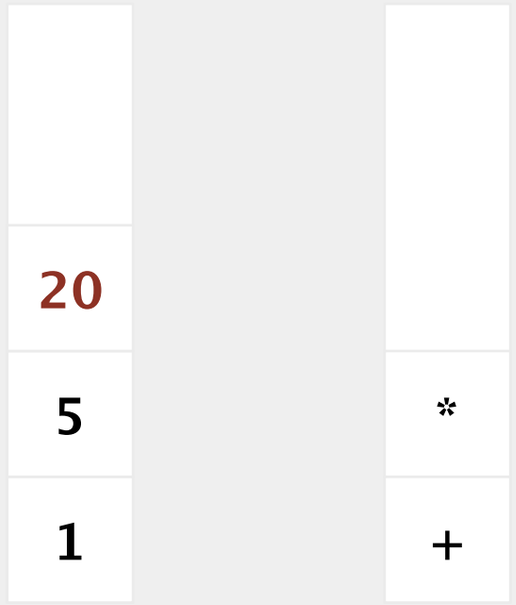
Dijkstra's two-stack algorithm

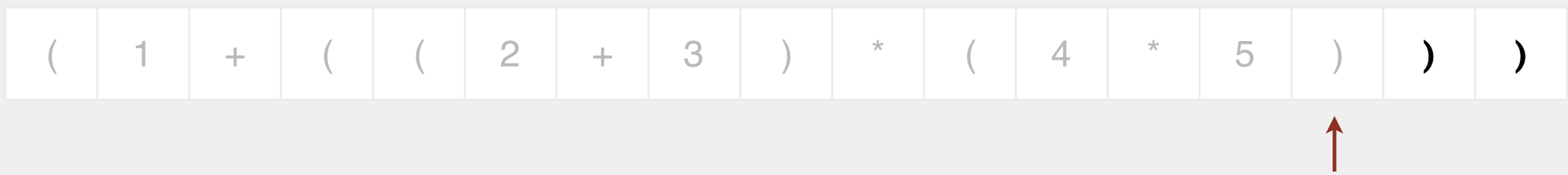
Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.





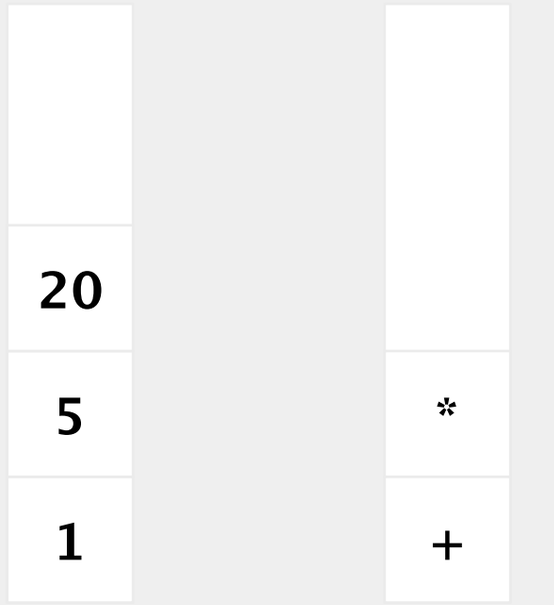
Dijkstra's two-stack algorithm

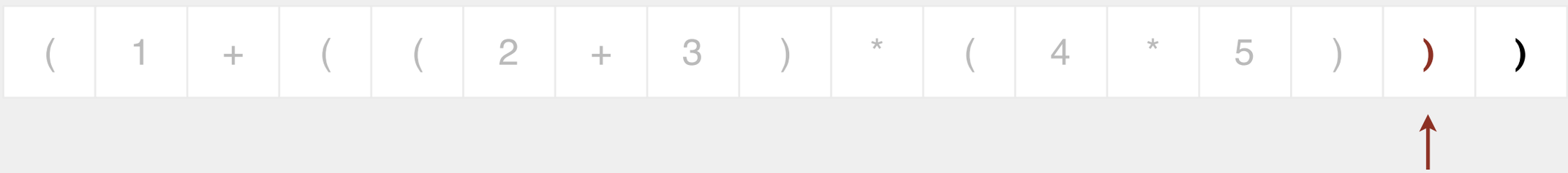
Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.





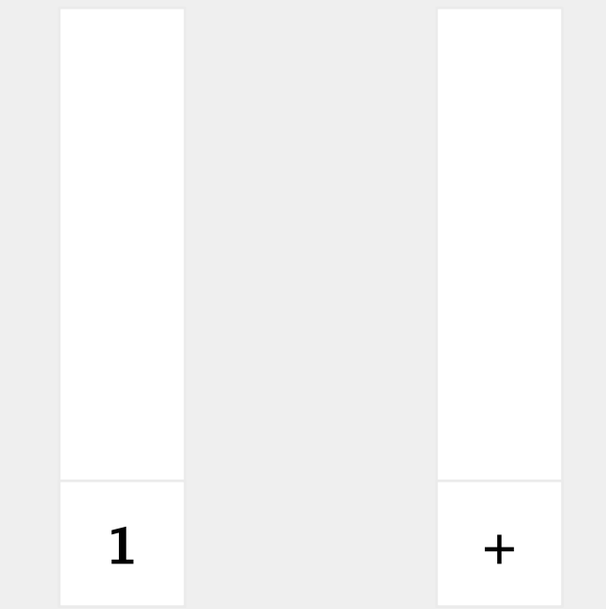
Dijkstra's two-stack algorithm

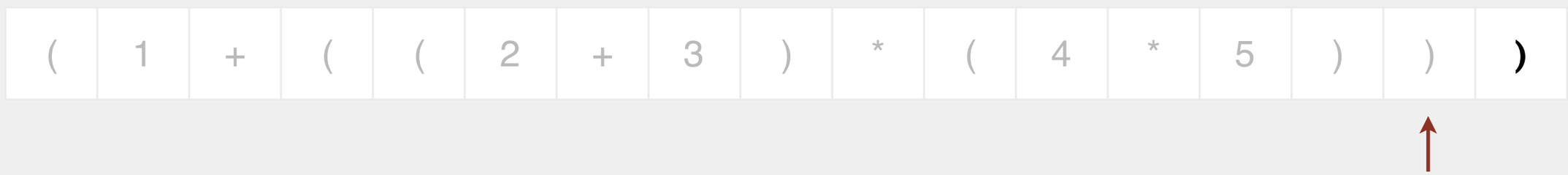
Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.





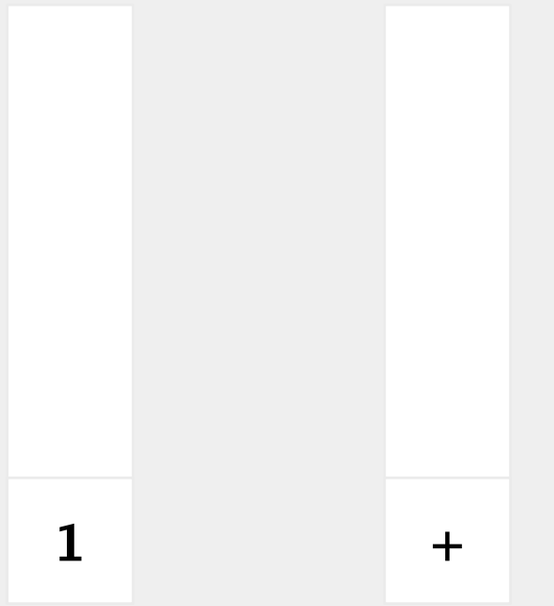
Dijkstra's two-stack algorithm

Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.



20	*	5	=	100

(	1	+	(	(	2	+	3	)	*	(	4	*	5	)	)	)

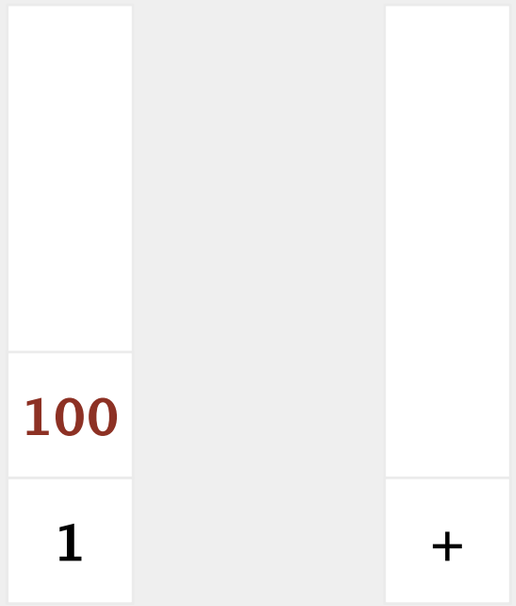
Dijkstra's two-stack algorithm

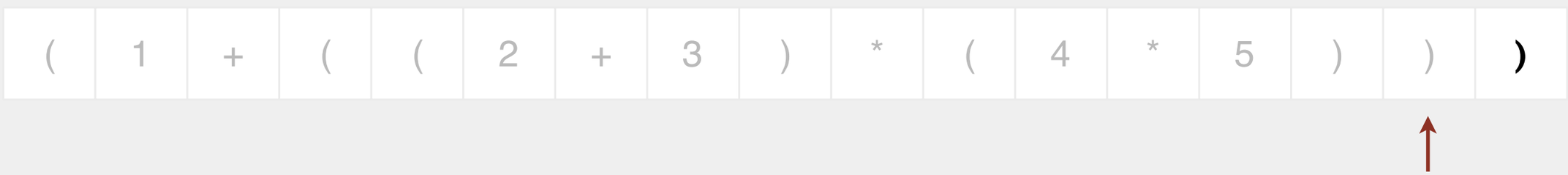
Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.





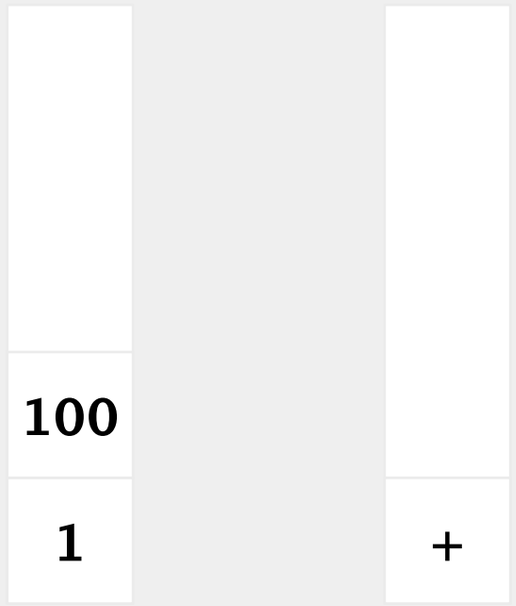
Dijkstra's two-stack algorithm

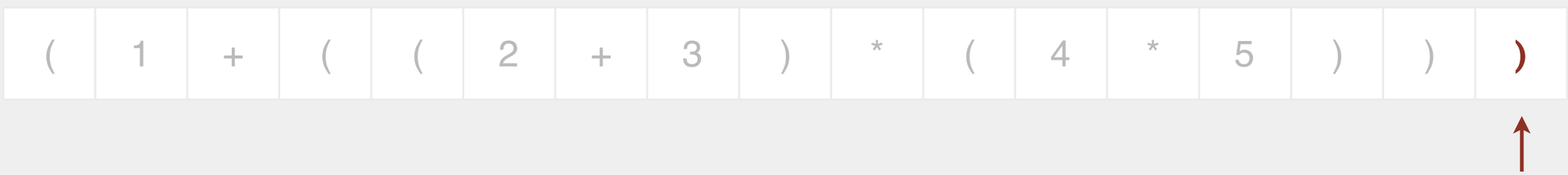
Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.





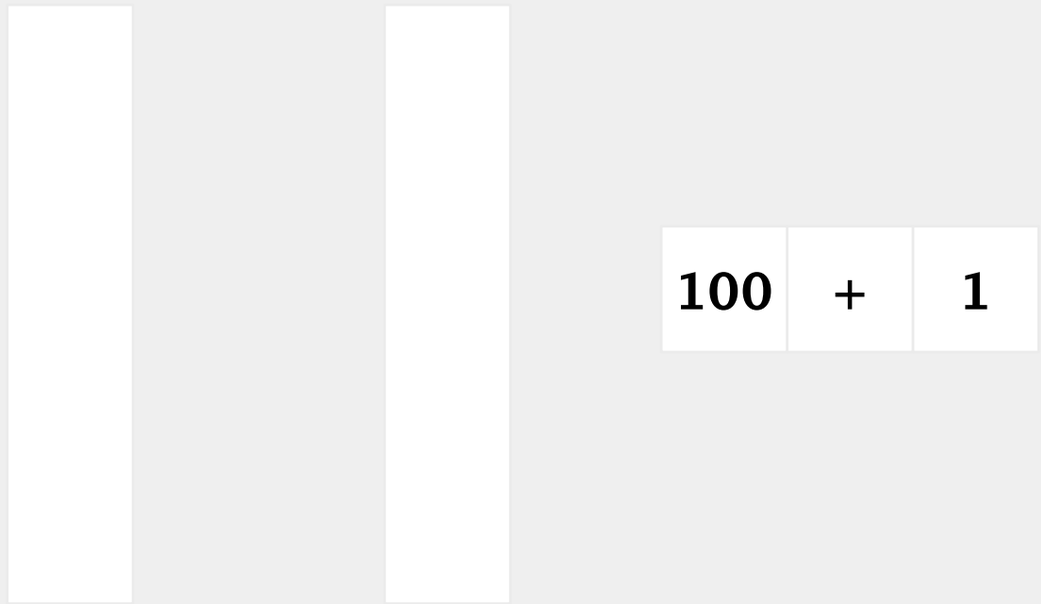
Dijkstra's two-stack algorithm

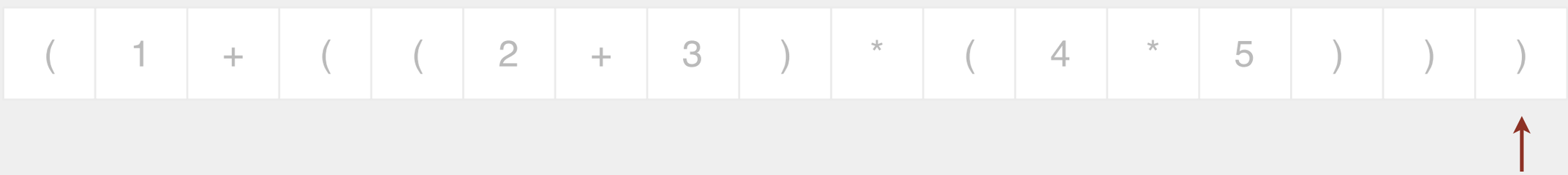
Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.





Dijkstra's two-stack algorithm

Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.

value stack operator stack

100	+	1	=	101

(	1	+	(	(	2	+	3	)	*	(	4	*	5	)	)	)

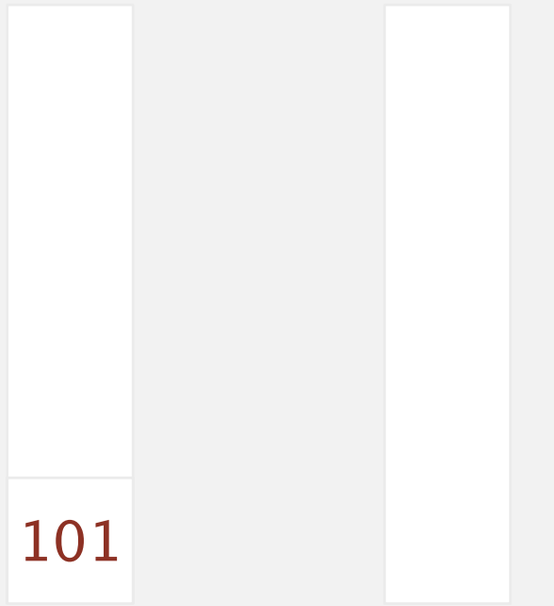
Dijkstra's two-stack algorithm

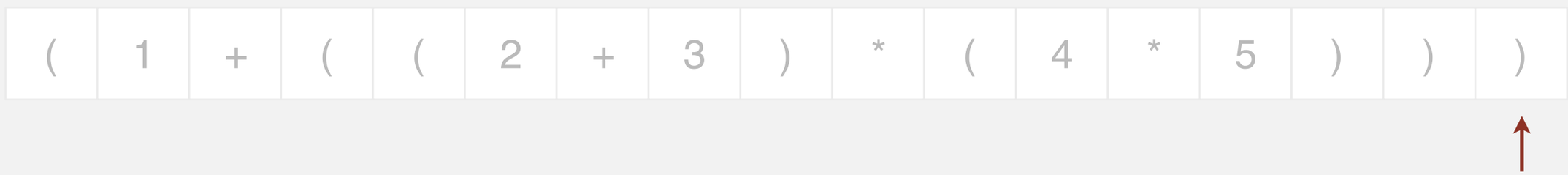
Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.





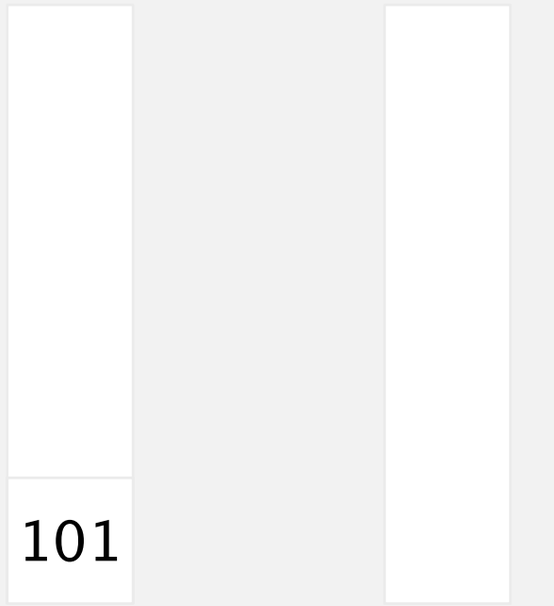
Dijkstra's two-stack algorithm

Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.



(	1	+	(	(	2	+	3	)	*	(	4	*	5	)	)	)

Dijkstra's two-stack algorithm

Value: push onto the value stack.

Operator: push onto the operator stack.

Left parenthesis: ignore.

Right parenthesis: pop operator and two values; push the result of applying that operator to those values onto the value stack.

End: if there are any operators remaining in the operator stack, then implement them.

result

(	1	+	(	(	2	+	3	)	*	(	4	*	5	)	)	)