Conversion of Prefix to Postfix

Data Structure
and Algorithm

Computer
Science

Stack

A stack is a linear data structure that follows the Last-In-First-Out (LIFO) principle. Think of it as a collection of items stacked on top of each other, similar to a stack of plates.

Key Operations

Prefix Notation

- Operators precede their operands. Example: "+ AB" represents "A + B".

Postfix Notation

- Operators follow their operands. Example: "AB +" represents "A + B".

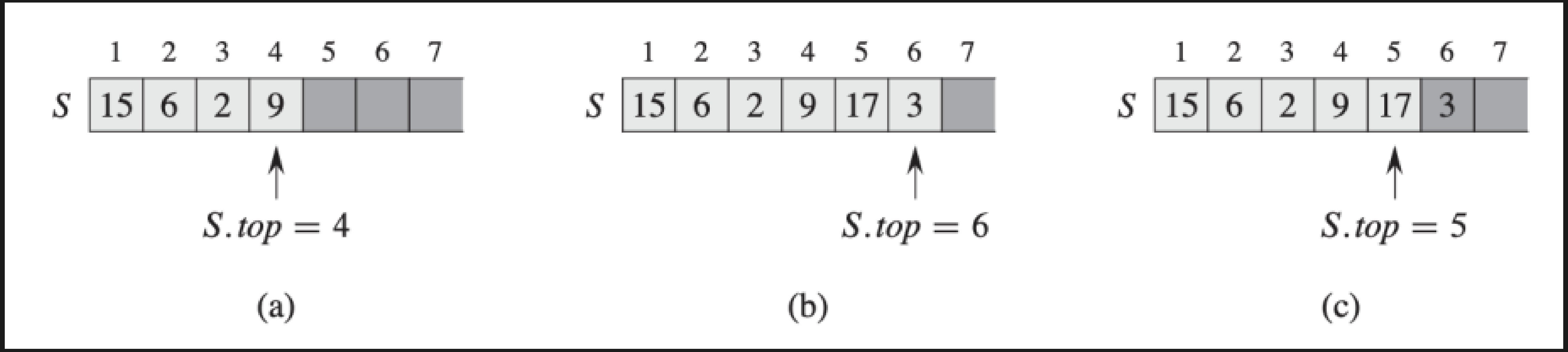
Key Operations

- Push: Adds an item to the top of the stack.
- Pop: Removes and returns the item from the top of the stack.
- Peek/Top: Retrieves the item from the top without removing it.
- Stacks are often used for managing function calls, expression evaluation, and more.

Stack

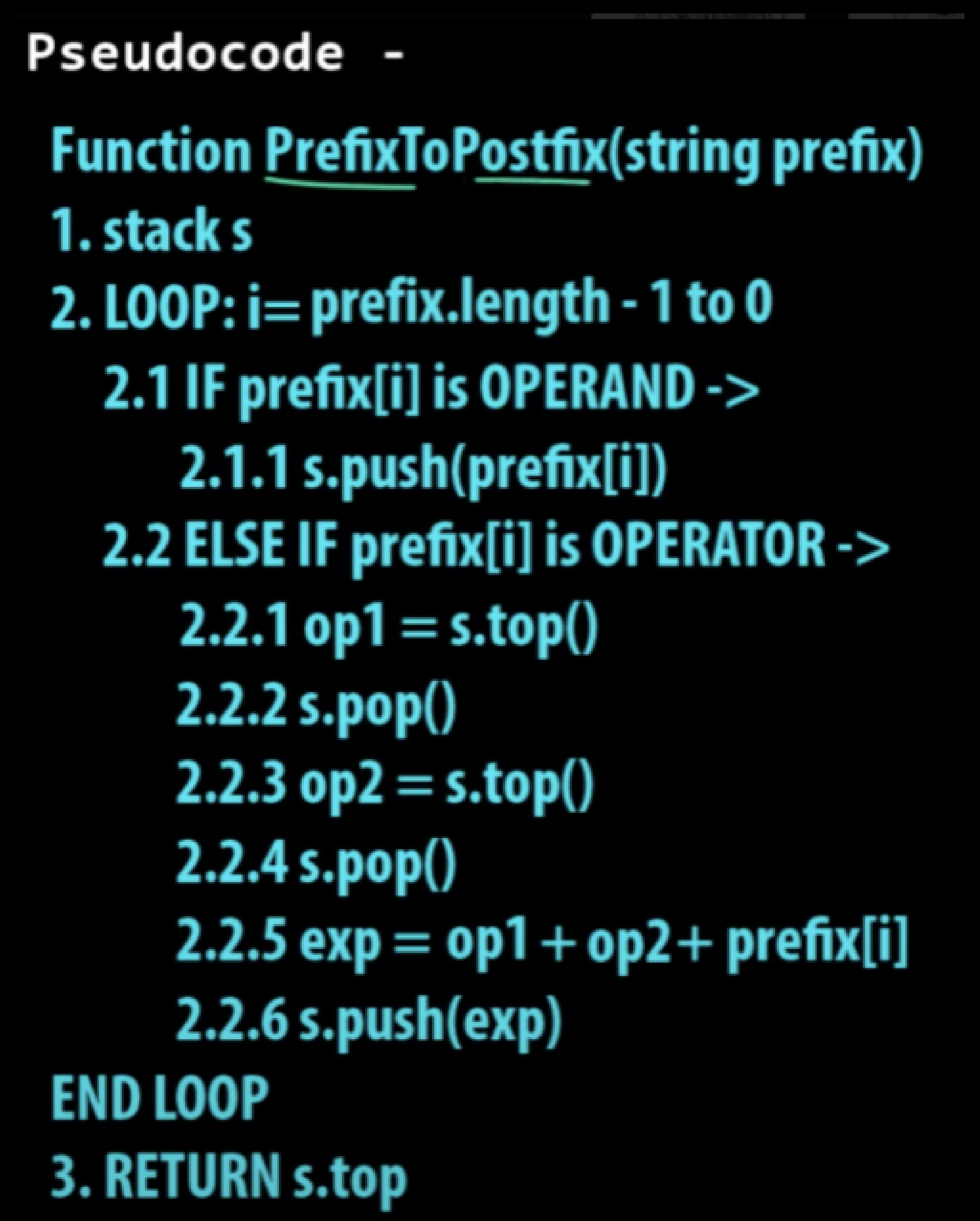
Insert operation on Stack is called Push, Delete operation

Using S.top



Pseudocode

PREFIX TO POSTFIX



```
Pseudocode -

Function PrefixToPostfix(string prefix)
1. stack s
2. LOOP: i= prefix.length - 1 to 0
   2.1 IF prefix[i] is OPERAND ->
       2.1.1 s.push(prefix[i])
   2.2 ELSE IF prefix[i] is OPERATOR ->
       2.2.1 op1 = s.top()
       2.2.2 s.pop()
       2.2.3 op2 = s.top()
       2.2.4 s.pop()
       2.2.5 exp = op1 + op2+ prefix[i]
       2.2.6 s.push(exp)
END LOOP
3. RETURN s.top
```

Python Code

```python
stack = []
operators = set(['+', '-', '*', '/', '^', '%'])

def PrefixToPostfix(s):
    stack = []
    s = s[::-1]
    for i in s:
        if i in operators:
            a = stack.pop()
            b = stack.pop()
            temp = a + " " + b + " " + i
            stack.append(temp)
        else:
            stack.append(i)
    return stack[0]

inputs = []
while True:
    line = input()
    line = line.replace(" ", "")
    if line == '0':
        break
    inputs.append(line)
results = []
for input_line in inputs:
    result = PrefixToPostfix(input_line)
    print(result)
    results.append(result)
```

Example

PEFIX TO POSTFIX

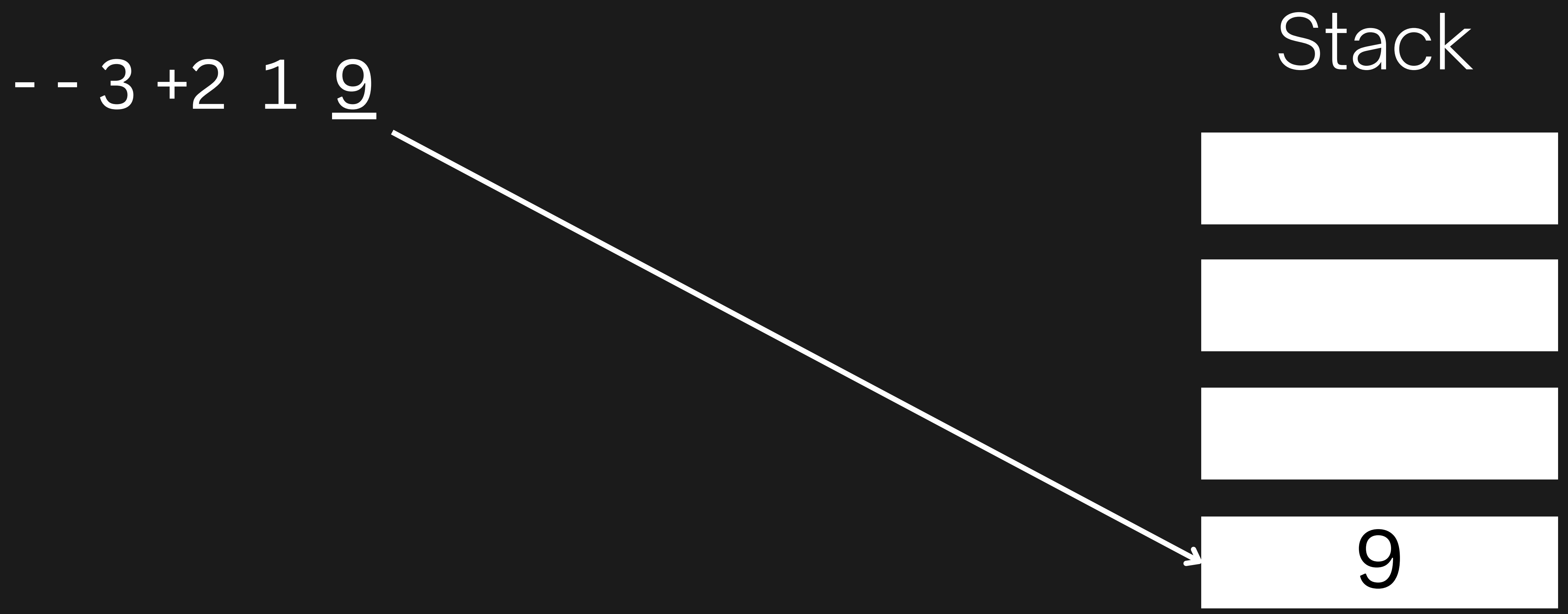
- **SCAN** from left to right
- **SELECT** first instance of 1 operators followed by 2 consecutive operands
- **CONVERT** it to postfix format
- **SUBSTITUTE** the sub postfix by 1 temporary operand variable
- **REPEAT** this process until the entire prefix expression is converted into postfix expression

PREFIX AND POSTFIX FORMART PREFIX: operator come before two operands (+ a b) POSTFIX: has two operands before the operator (a b +)

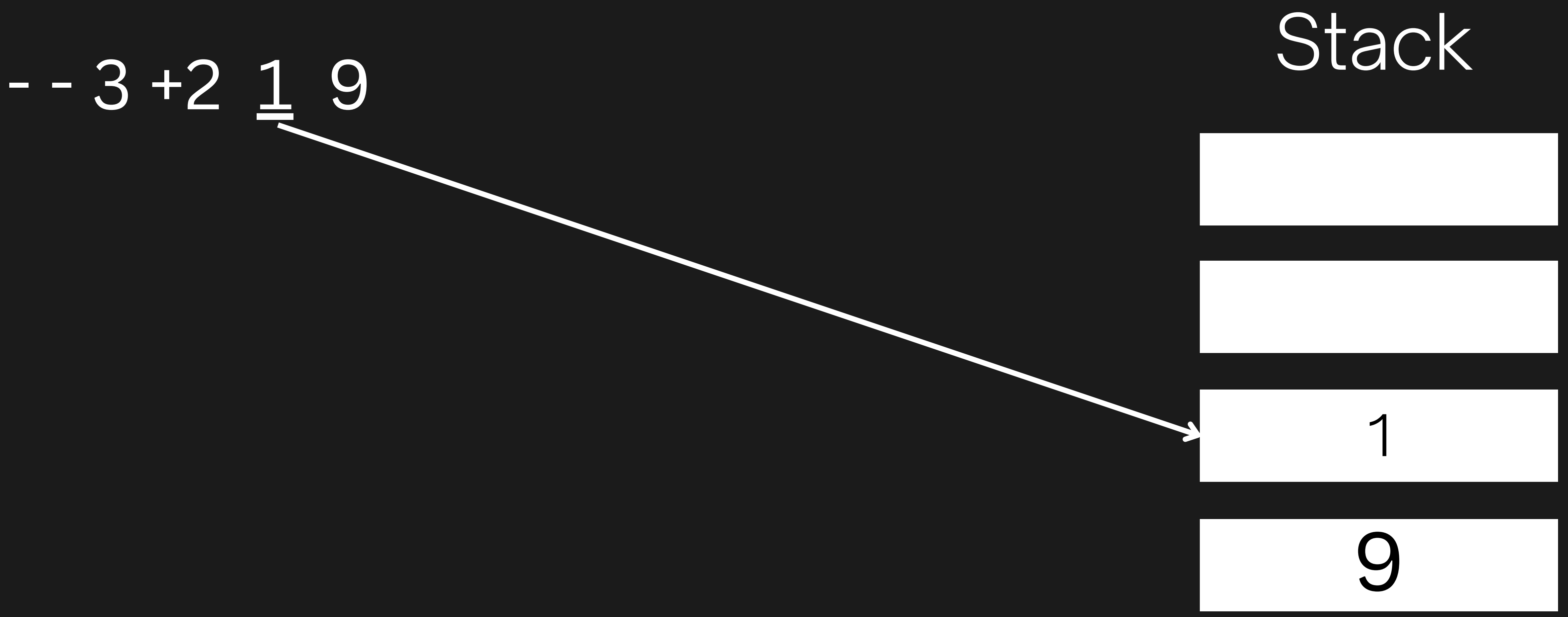
The input is + - a b c

1ST STEP SCAN		2ND STEP SELECT		3RD STEP CONVERT		4RD STEP SUBSTITUTE
+ - a b c	➡	**-, a, b**	➡	**a b -**	➡	**x1 = a b -**
+ x1 c	➡	**+, x1, c**	➡	**x1 c +**	➡	**a b - c +**

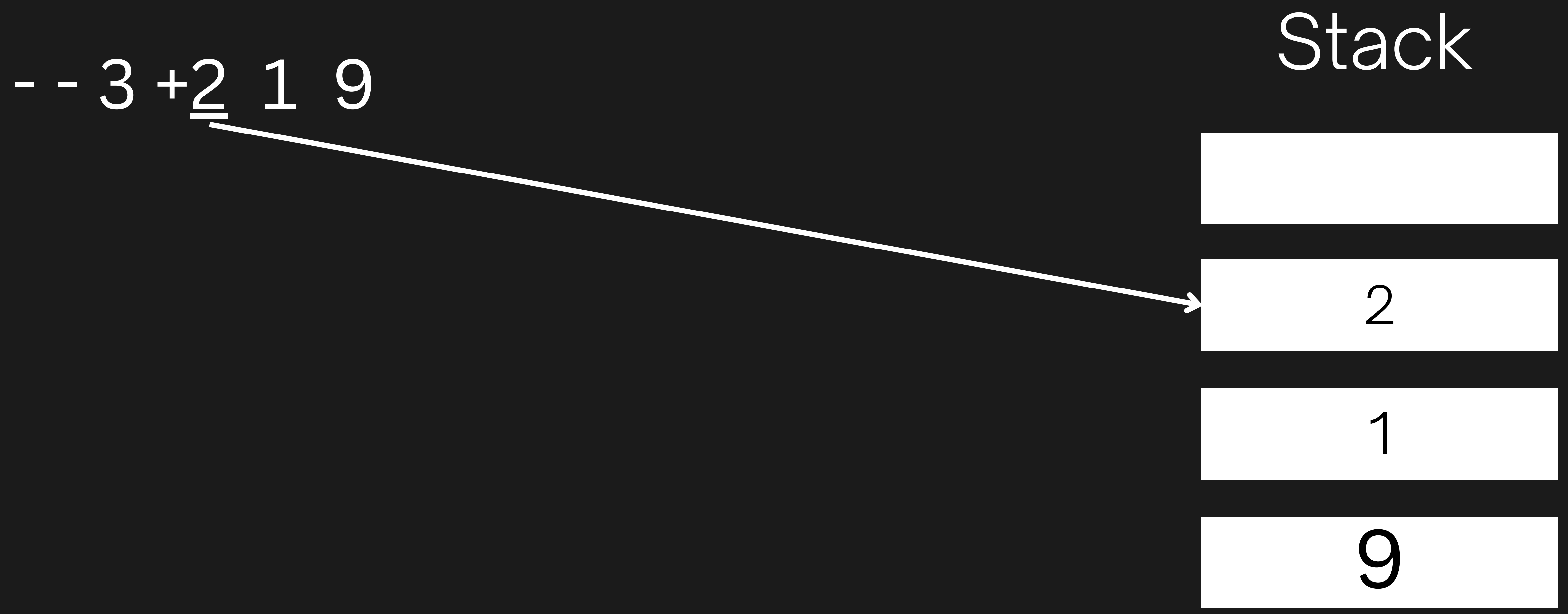
(1) Prefix expression



(1) Prefix expression

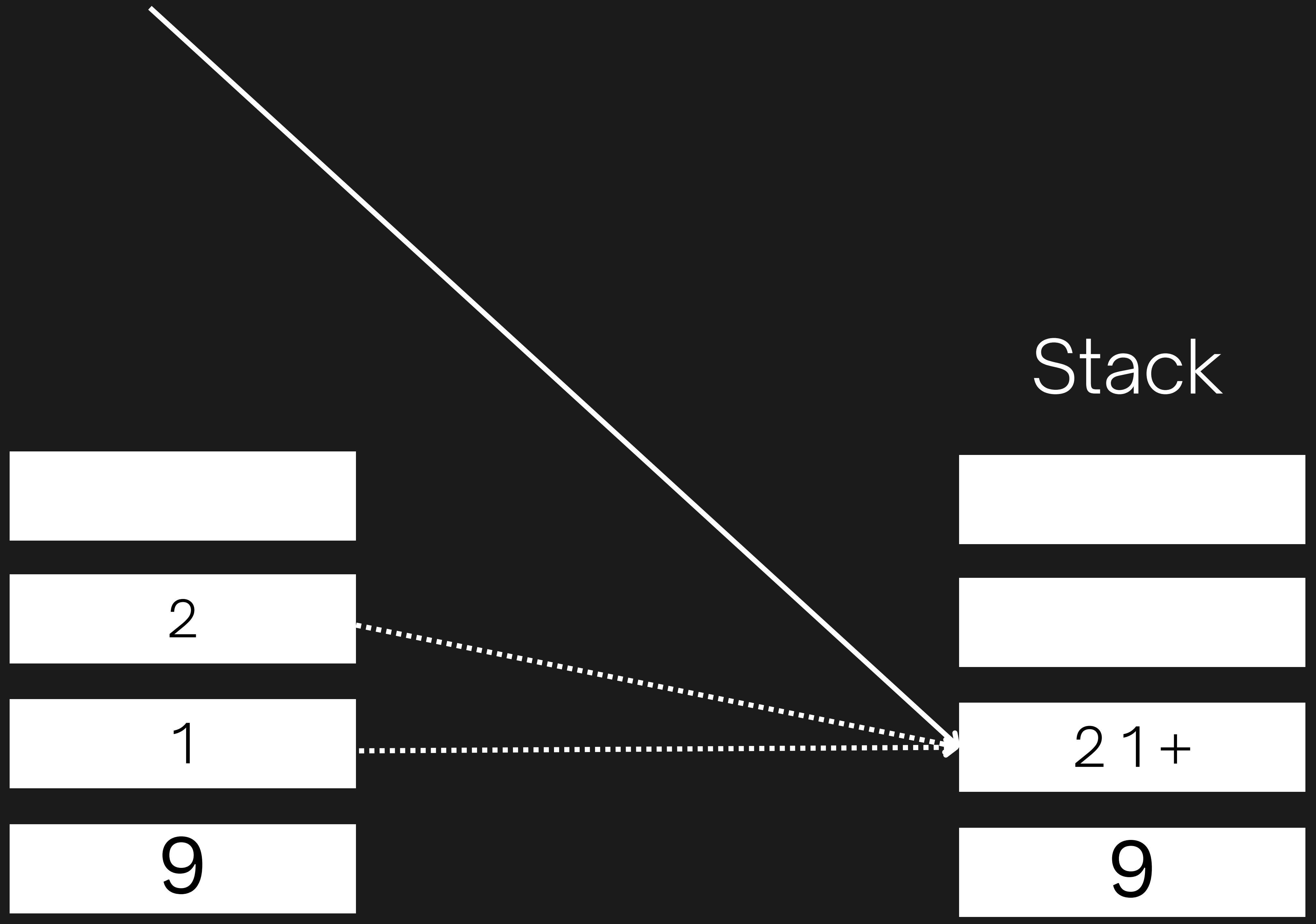


(1) Prefix expression



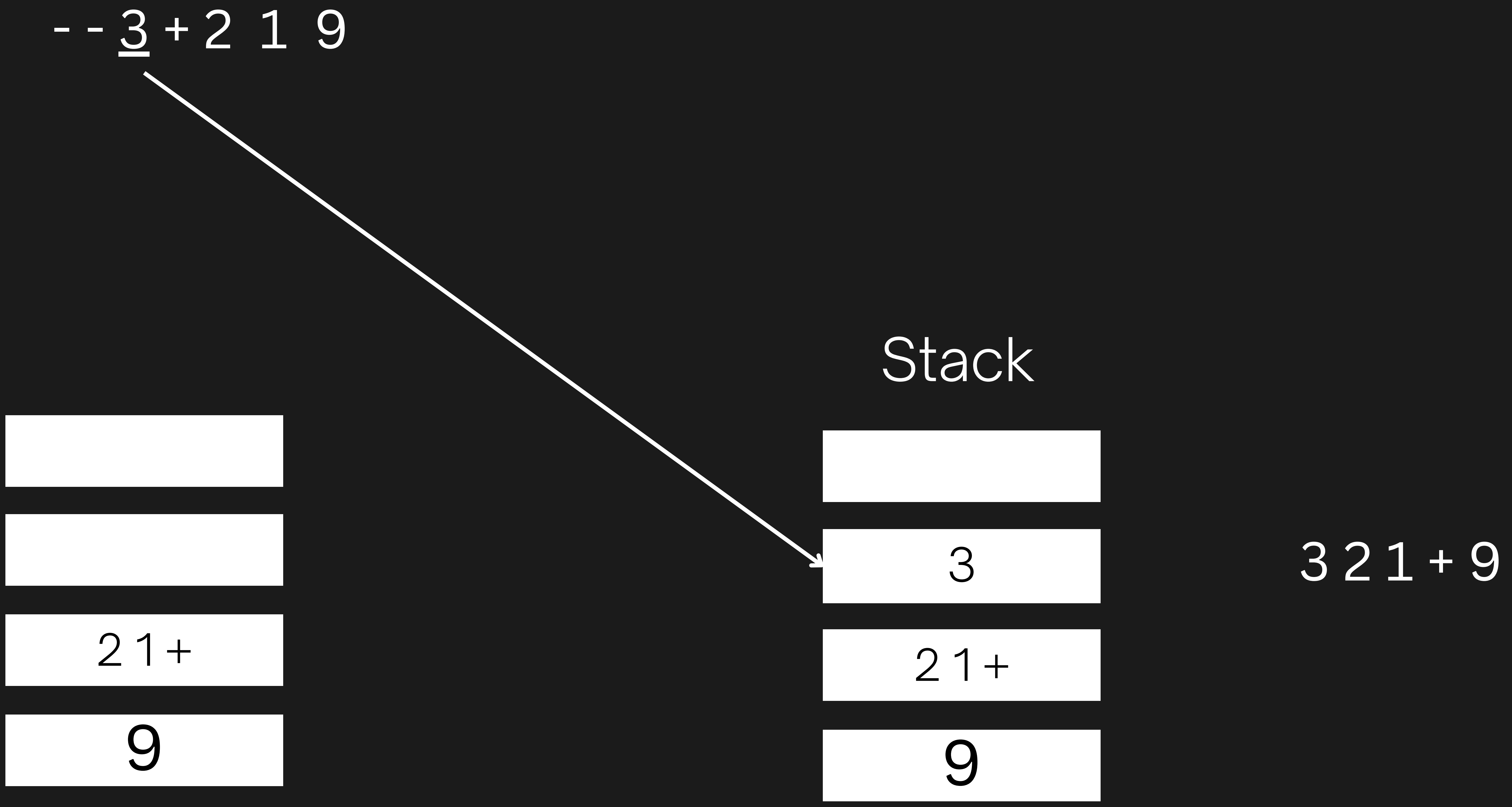
(1) Prefix expression

- - 3 + 2 1 9

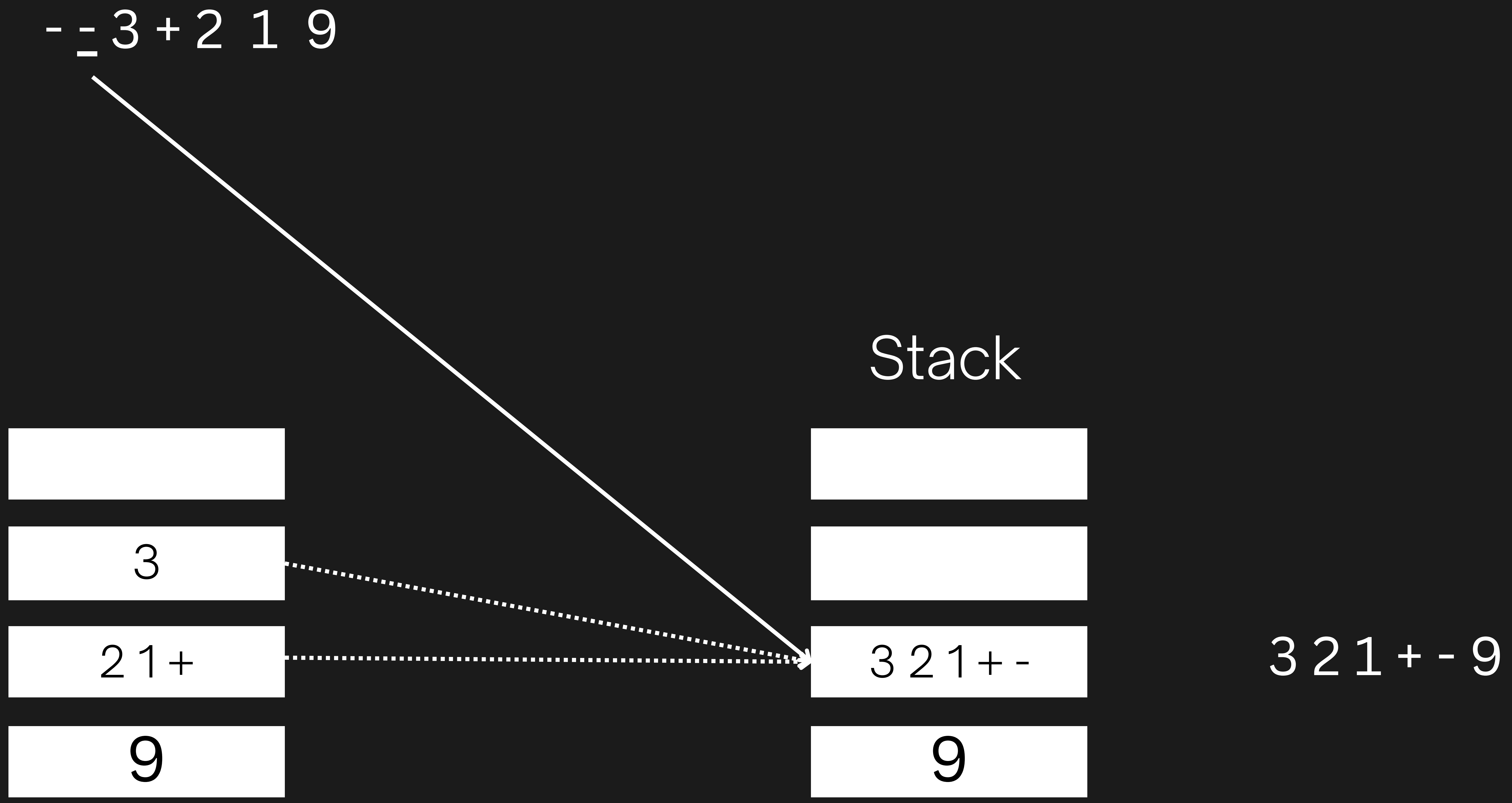


2 1 + 9

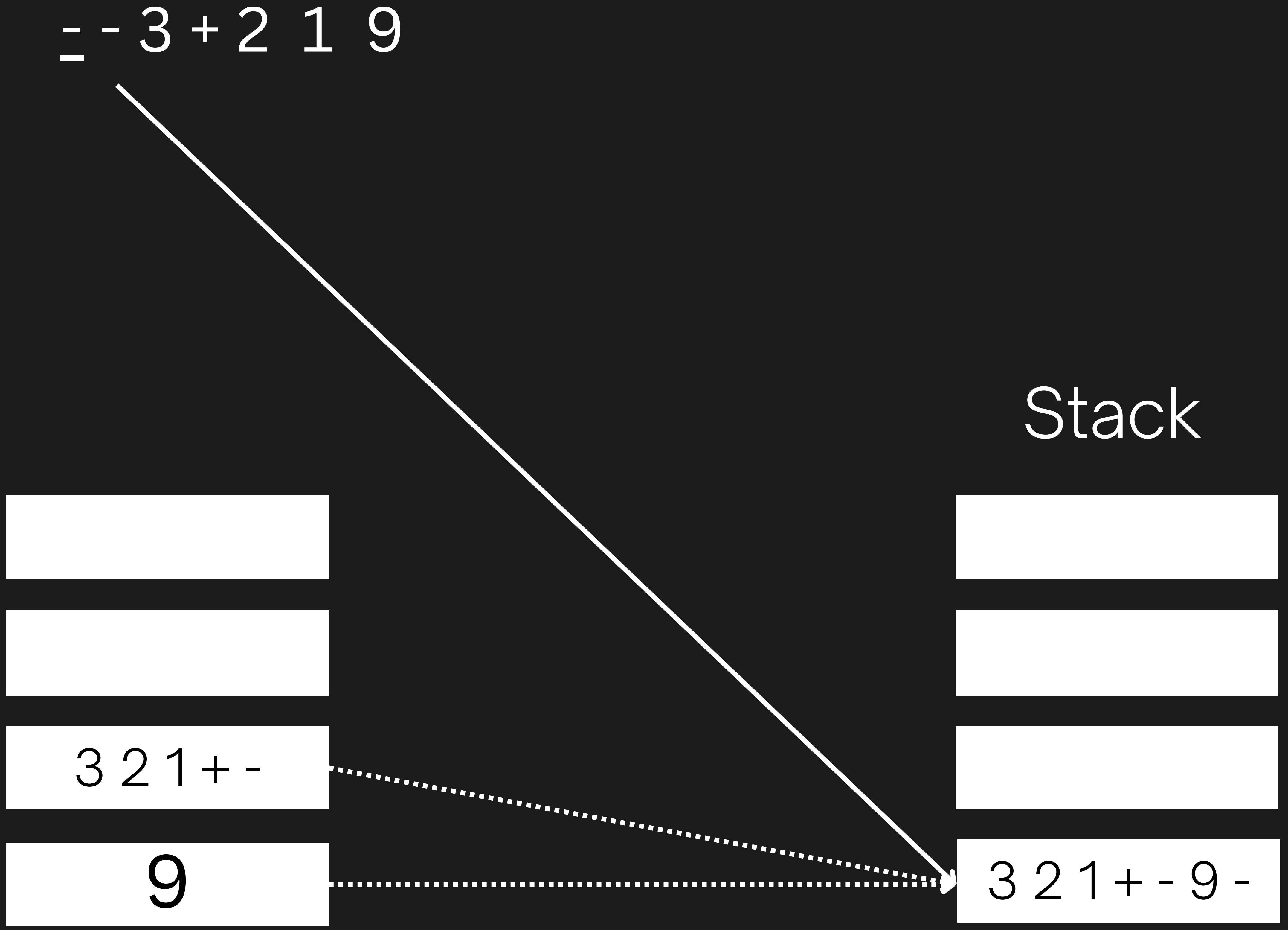
(1) Prefix expression



(1) Prefix expression



(1) Prefix expression



3 2 1 + - 9 -