# Basic Calculator

For this problem it requires parsing and evaluating an arithmetic expression with +, − and parentheses efficiently.

## Key Idea:

We handle nested parentheses and signs using a stack:

1. Iterate through the string:
   - Digit: Build the full number.
   - + or −: Update sign (1 or −1).
   - (: Push current result and sign onto stack (for later restoration), then reset.
   - ): Pop sign and previous result, combine with current.
2. Add sign * number to result when a number ends or at the end.

## Handling Unary:

- Unary minus is valid (e.g., −(2+3)).
- When − appears:
  - If previous token was ( or start of string, treat it as unary (e.g. prepend 0).

## Algorithm:

1. Initialize:
   - result = 0
   - sign = 1
   - stack = []
2. Traverse string s:
   - If digit → parse full number, add sign * num to result.
   - If + → sign = 1
   - If − → sign = −1
   - If ( → push result and sign on stack, reset result = 0, sign = 1
   - If ) → pop sign, pop previous result, combine.

## Complexity:

- Time: $O(n)$ – single pass through string
- Space: $O(n)$ – stack for nested parentheses