

439. Ternary Expression Parser

Description

Given a string `expression` representing arbitrarily nested ternary expressions, evaluate the expression, and return *the result of it*.

You can always assume that the given expression is valid and only contains digits, `'?'`, `':'`, `'T'`, and `'F'` where `'T'` is true and `'F'` is false. All the numbers in the expression are **one-digit** numbers (i.e., in the range `[0, 9]`).

The conditional expressions group right-to-left (as usual in most languages), and the result of the expression will always evaluate to either a digit, `'T'` or `'F'`.

Example 1:

Input: `expression = "T?2:3"`
Output: `"2"`
Explanation: If true, then result is 2; otherwise result is 3.

Example 2:

Input: `expression = "F?1:T?4:5"`
Output: `"4"`
Explanation: The conditional expressions group right-to-left. Using parenthesis, it is read/evaluated as:
`"(F ? 1 : (T ? 4 : 5))"` --> `"(F ? 1 : 4)"` --> `"4"`
or `"(F ? 1 : (T ? 4 : 5))"` --> `"(T ? 4 : 5)"` --> `"4"`

Example 3:

Input: `expression = "T?T?F:5:3"`
Output: `"F"`
Explanation: The conditional expressions group right-to-left. Using parenthesis, it is read/evaluated as:
`"(T ? (T ? F : 5) : 3)"` --> `"(T ? F : 3)"` --> `"F"`
`"(T ? (T ? F : 5) : 3)"` --> `"(T ? F : 5)"` --> `"F"`

Constraints:

- `5 <= expression.length <= 104`
- `expression` consists of digits, `'T'`, `'F'`, `'?'`, and `':'`.
- It is **guaranteed** that `expression` is a valid ternary expression and that each number is a **one-digit number**.

