Given a boolean `expression`with the following symbols:

- `T` - `true`
- `F` - `false`

and the following operators between the symbols:

- `&` - boolean `AND`
- `|` - boolean `OR`
- `^` - boolean `XOR`

Count the number of ways that you can parenthesize the expression so that the expression evaluates to `true`, and return this answer modulo `1003`.

Example

- For `expression = "T&T|F^F"`, the output should be `booleanParenthesization(expression) = 5`.

  Here are all of the possible combinations:

  - `((T&T)|(F^F))`
  - `(T&((T|F)^F))`
  - `(((T&T)|F)^F)`
  - `((T&(T|F))^F)`
  - `(T&(T|(F^F)))`

- For `expression = "F|T^F"`, the output should be `booleanParenthesization(expression) = 2`.

  Here are all of the possible combinations:

  - `(F|(T^F))`
  - `((F|T)^F)`

Input/Output

- **[time limit] 4000ms (py3)**
- **[input] string expression**

  Guaranteed constraints:
  `1 ≤ expression.length ≤ 100`.

- **[output] integer**

  The number of ways that you can parenthesize the expression modulo `1003`.