# 1106. Parsing A Boolean Expression

## Description

A **boolean expression** is an expression that evaluates to either `true` or `false`. It can be in one of the following shapes:

- `'t'` that evaluates to `true`.
- `'f'` that evaluates to `false`.
- `'!(subExpr)'` that evaluates to **the logical NOT** of the inner expression `subExpr`.
- `'&(subExpr_1, subExpr_2, ..., subExpr_n)'` that evaluates to **the logical AND** of the inner expressions `subExpr_1, subExpr_2, ..., subExpr_n` where `n >= 1`.
- `'|(subExpr_1, subExpr_2, ..., subExpr_n)'` that evaluates to **the logical OR** of the inner expressions `subExpr_1, subExpr_2, ..., subExpr_n` where `n >= 1`.

Given a string `expression` that represents a **boolean expression**, return *the evaluation of that expression*.

It is **guaranteed** that the given expression is valid and follows the given rules.

**Example 1:**

```
Input: expression = "&(|(f))"
Output: false
Explanation: 
First, evaluate |(f) --> f. The expression is now "&(f)".
Then, evaluate &(f) --> f. The expression is now "f".
Finally, return false.
```

**Example 2:**

```
Input: expression = "|(f,f,f,t)"
Output: true
Explanation: The evaluation of (false OR false OR false OR true) is true.
```

**Example 3:**

```
Input: expression = "!(&(f,t))"
Output: true
Explanation: 
First, evaluate &(f,t) --> (false AND true) --> false --> f. The expression is now "!(f)".
Then, evaluate !(f) --> NOT false --> true. We return true.
```

**Constraints:**

- $1 <= \texttt{expression.length} <= 2 * 10^4$
- expression[i] is one following characters: `'('`, `')'`, `'&'`, `'|'`, `'!'`, `'t'`, `'f'`, and `','`.