

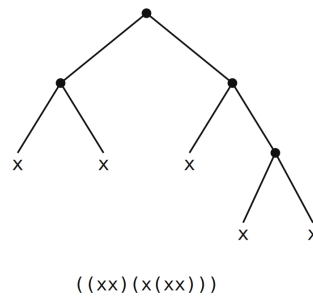
Binary Rotations

Input file: *standard input*
 Output file: *standard output*
 Time limit: 7 seconds
 Memory limit: 1024 mebibytes

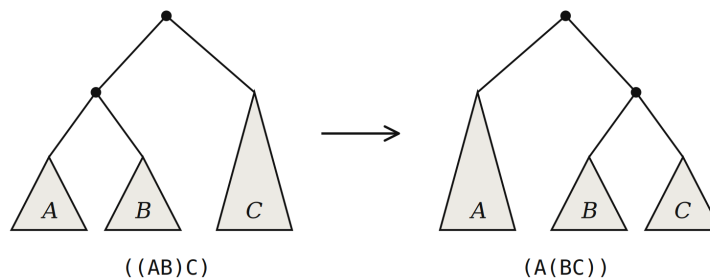
A *valid expression* is defined recursively:

- x is a valid expression;
- if A and B are valid expressions, then (AB) is also a valid expression.

For example, $((xx)(x(xx)))$ is a valid expression with five occurrences of x . Every valid expression corresponds to a binary tree: each x is a leaf, and each pair of brackets is an internal node whose children are the two subexpressions it encloses.



In one operation, you may replace a subexpression of the form $((AB)C)$ with $(A(BC))$, where A , B , and C are arbitrary valid expressions. In the tree representation, this rotates a single edge.



We say that an expression u can be obtained from an expression v if u can be produced from v by applying zero or more operations. In particular, every expression can be obtained from itself.

You are given an integer ℓ . The number of valid expressions with exactly ℓ occurrences of x is the Catalan number

$$n = C_{\ell-1} = \frac{1}{\ell} \binom{2\ell-2}{\ell-1},$$

and all of them are enumerated in lexicographical order as $S_1, S_2, \dots, S_n$, using the character order ' $(< x <)$ '. Here, $\binom{a}{b} = \frac{a!}{b!(a-b)!}$ is the binomial coefficient.

You are also given a permutation $p_1, p_2, \dots, p_n$ of the integers from 1 to n ; the value p_i represents the expression S_{p_i} . Count the pairs (i, j) with $1 \leq i < j \leq n$ such that S_{p_i} can be obtained from S_{p_j} .

Input

The first line contains one integer ℓ ($1 \leq \ell \leq 14$).

The second line contains $n = C_{\ell-1}$ integers $p_1, p_2, \dots, p_n$: a permutation of $1, 2, \dots, n$.

Output

Print one integer: the number of pairs (i, j) such that $1 \leq i < j \leq n$ and S_{p_i} can be obtained from S_{p_j} .

Example

<i>standard input</i>	<i>standard output</i>
4 3 1 5 2 4	3

Note

For $\ell = 4$ there are $n = C_3 = 5$ valid expressions, in lexicographical order: $S_1 = ((\mathbf{x}\mathbf{x})\mathbf{x})\mathbf{x}$, $S_2 = ((\mathbf{x}(\mathbf{x}\mathbf{x}))\mathbf{x})$, $S_3 = ((\mathbf{x}\mathbf{x})(\mathbf{x}\mathbf{x}))$, $S_4 = (\mathbf{x}((\mathbf{x}\mathbf{x})\mathbf{x}))$, $S_5 = (\mathbf{x}(\mathbf{x}(\mathbf{x}\mathbf{x})))$.

A single operation turns S_1 into S_2 or into S_3 , turns S_2 into S_4 , and turns each of S_3 and S_4 into S_5 ; no operation applies to S_5 .

The three pairs that count are $(i, j) = (1, 2)$, since S_3 can be obtained from S_1 ; $(3, 4)$, since S_5 can be obtained from S_2 ; and $(3, 5)$, since S_5 can be obtained from S_4 .