Baltic OI '12 P1 - Brackets

Time limit: 1.0s **Memory limit:** 64M

Baltic Olympiad in Informatics: 2012 Day 1, Problem 1

Let's define a correct string of brackets as follows:

- `()` and `[]` are correct strings of brackets;
- if `A` is a correct string of brackets, then `(A)` and `[A]` are also correct strings of brackets;
- if `A` and `B` are both correct strings of brackets, then the concatenation `AB` is also a correct string of brackets;

In a correct string of brackets which contains at least one pair of square brackets: `[` and corresponding `]`, each square bracket (both opening and closing) was replaced by the **opening** round bracket, therefore obtaining a *broken string of brackets*.

For example, `((` and `(((((()))` both are broken strings of brackets. First string is obtained from the correct strings of brackets `[]`. Second string may be obtained only from the following four correct strings of brackets: `[](((())))`, `([](()))`, `(([](()))` or `((([])))`.

Your task is for a given broken string of brackets calculate the number of possible correct strings of brackets from which the given broken string may have been obtained.

Constraints

$2 \leq N \leq 3 \times 10^4$

N is even.

Subtask 1 [20%]

$2 \leq N \leq 50$

Subtask 2 [45%]

$2 \leq N \leq 1\,000$

Subtask 3 [35%]

No additional constraints.

Input Specification

The first line of input contains an even integer N - the length of the given broken string of brackets. The second line contains N characters `(` and `)` - the given broken string of brackets.

Output Specification

Output a single integer - the required number of correct strings of brackets. Because the number of correct strings of brackets may be quite large, you should output the answer modulo $10^9 + 9$.

Sample Input 1

```
4
(((()
```

Sample Output 1

```
2
```

Explanation for Sample 1

The correct string of brackets are:

- `[]()`
- `([])`

Sample Input 2

```
8
(((((((((
```

Sample Output 2

```
14
```

Explanation for Sample 2

The correct string of brackets are:

- `[][][][]`
- `[[]][][]`
- `[[]][[]]`
- `[][][[]]`

- `[[[]]][]`
- `[[][]][]`
- `[][[][]]`
- `[][[[]]]`
- `[[[[]]]]`
- `[[][[]]]`
- `[[[]][]]`
- `[[][][]]`
- `[[[][]]]`
- `[][[]][]`