C. Even Positions

time limit per test: 2 seconds
memory limit per test: 256 megabytes

Monocarp had a regular bracket sequence s of length n (n is even). He even came up with his own way to calculate its cost.

He knows that in a regular bracket sequence (RBS), each opening bracket is paired up with the corresponding closing bracket. So he decided to calculate the *cost* of RBS as the sum of distances between pairs of corresponding bracket pairs.

For example, let's look at RBS `(())()`. It has three pairs of brackets:

- `(__)__`: the distance between brackets at position 1 and at 4 is $4 - 1 = 3$;
- `_()___`: the distance is $3 - 2 = 1$;
- `_____()`: the distance is $6 - 5 = 1$.

So the cost of `(())()` is $3 + 1 + 1 = 5$.
Unfortunately, due to data corruption, Monocarp lost all characters on odd positions $s_1, s_3, \ldots, s_{n-1}$. Only characters on even positions $(s_2, s_4, \ldots, s_n)$ remain. For example, `(())()` turned to `_(_)_`.

Monocarp wants to restore his RBS by placing brackets on the odd positions. But since the restored RBS may not be unique, he wants to choose one with **minimum cost**. It's too hard to do for Monocarp alone, so can you help him?

Reminder: A *regular bracket sequence* is a string consisting of only brackets, such that this sequence, when inserted `1`-s and `+`-s, gives a valid mathematical expression. For example, `()`, `(())` or `(()())()` are RBS, while `)`, `()(` or `())(()` are not.

Input

The first line contains a single integer t ($1 \le t \le 5000$) — the number of test cases. Next t cases follow.

The first line of each test case contains a single integer n ($2 \le n \le 2 \cdot 10^5$; n is even) — the length of string s.

The second line of each test case contains a string s of length n, where all characters on the odd positions are '_' and all characters on the even positions are either '(' or ')'.

Additional constraints:

- s can be restored to at least one regular bracket sequence;
- the total sum of n over all test cases doesn't exceed $2 \cdot 10^5$.

Output

For each test case, print one integer — the minimum cost of the regular bracket sequence that can be obtained from s by replacing '_'-s with brackets.

Example

input	Copy
4	

```
6
_(_)_)
2
_)
8
_)_)_)_)
8
_(_)_(_)
```

output Copy

```
5
1
4
8
```

Note

In the first test case, it's optimal to make s equal to `(())()`. The cost of s will be equal to $3 + 1 + 1 = 5$.

In the second test case, the only option is to make s equal to `()` with cost 1.

In the third test case, the only possible RBS is `()()()()` with cost $1 + 1 + 1 + 1 = 4$.

In the fourth test case, it's optimal to make s equal to `(())(())` with cost $3 + 1 + 3 + 1 = 8$.