Matter of Parentheses (`parentheses`)

Figure 1: Brackets. Why not?

Antonio loves brackets (or parentheses). He calls a sequence of '`(`' and '`)`' characters a "valid bracket sequence" if either:

- It's an empty sequence, or
- For each open bracket we can find a closed bracket on its right such that the substring between these two brackets forms a valid bracket sequence, and such that after removing that substring the remaining brackets also form a valid bracket sequence.

For example `(())` and `()()` are both valid, while `)(` and `())(()` are not.

Furthermore, he calls a string of parentheses "K-valid" if, after adding K open brackets to its left and K closed brackets to its right, it is valid (either because it was valid before, or because it became valid).

For example `)(` and `())(()` are both 1-valid, while `))((` is 2-valid.

Antonio is curious to find for any given N and K how many bracket sequences there are with length $2N$ that are K-valid. Because the result may be very large, Antonio is only interested in its remainder after dividing it by 1 000 000 007.

Help Antonio by writing a program that quickly calculates this number for Q different queries.

Input

The first line contains one integer Q, the number of queries Antonio is interested in.

The next N lines contain each a pair of integers: N_i and K_i.

Output

You need to write Q lines containing each the result of the i-th query: for each query, print the number of strings of length $2N_i$ that are K_i-valid, modulo 1 000 000 007.

Constraints

- $1 \leq Q \leq 100\,000$.
- $1 \leq N_i \leq 1\,000\,000$.
- $0 \leq K_i \leq 1\,000\,000$.

Examples

input	output

```
3
2 1
4 2
5 3
```
	```
	5
	62
	242
	```

Explanation

In the first query there are 5 strings of length 4 that are 1-valid. They are highlighted below:

- In `( (()) )` the sequence `(())` is valid, but also 1-valid.
- In `( ()() )` the sequence `()()` is valid, but also 1-valid.
- In `( ())( )` the sequence `())(` is invalid, but 1-valid.
- In `( )(() )` the sequence `)(()` is invalid, but 1-valid.
- In `( )()( )` the sequence `)()(` is invalid, but 1-valid.