

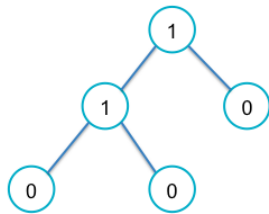
Counting Trees (prob7)

You own your company, UberTries Solutions. One of your clients wants to represent binary trees in a compressed format. He proposes an ingenious method of compressing binary trees by performing an in-order traversal, and saving the resulting string.

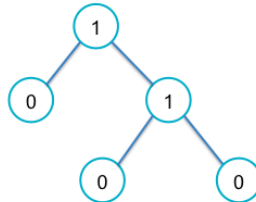
You being the keen problem solver, notice that his method has errors. In fact, you notice that there are multiple binary trees that you can recreate from the same string representation. Your task is to find the total ^[1] number of such valid binary trees.

You will be given a rooted binary tree, which means that the tree has a root node, and each internal node can have either *one* or *two* children. The internal nodes are numbered 1s, and leaves are numbered 0s. You will be given a string representation of the binary tree. The string representation is a result of performing the standard in-order traversal of the given tree. An outline of an in-order traversal is given below.

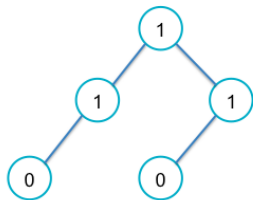
```
in-order-traversal(node)
  if node is leaf then
    print(0)
  else
    if node.left exists then
      in-order-traversal(node.left)
    end if
    print(1)
    if node.right exists then
      in-order-traversal(node.right)
    end if
  end if
```



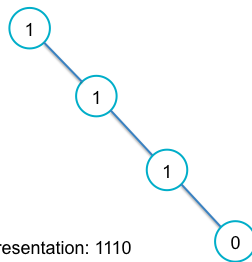
i) Representation: 01010



ii) Representation: 01010



iii) Representation: 01101



iv) Representation: 1110

Figures i, ii, iii and iv shows examples of binary trees and its string representation. Note that internal nodes can have *one* child too. Also, notice that tree (i) and tree (ii) share the same string representation. In fact, for the string representation 01010, there are only 2 possible binary trees.

Input

The first line contains the number of test cases, N ($1 \leq N \leq 100$). In each of the following N lines, there will be a string S ($1 \leq \text{length of } S \leq 100$) consisting of 1s and 0s only.

Output

For each string S , output the number of possible valid trees modulo ^[2] 1,000,001 (one million and one) in a single line.

Sample Input

```
5
0
010
01010
0110010
1
```

Sample Output

```
1
1
2
0
0
```

Notes:

- a) Beware of integer overflow.
- b) Beware of run time limit exceeded.

^[1] Not the total but the total modulo 1,000,001.

^[2] $A \text{ modulo } N = A \% N$ (in Java, C++, C #, Python).