D - Partitioning by Palindrome

Source file name: `palindrome.py`
Time limit: 1 second

We say a sequence of characters is a *palindrome* iff it is the same written forwards and backwards. For example, `racecar` is a palindrome and `fastcar` is not.

A partition of a sequence of characters is a list of one or more disjoint non-empty groups of consecutive characters whose concatenation yields the initial sequence. For example, (`race`, `car`) is a partition of `racecar` into two groups.

Given a sequence of characters, we can always create a partition of these characters such that each group in the partition is a palindrome! Given this observation it is natural to ask: what is the minimum number of groups needed for a given string such that every group is a palindrome?

For example:

- `racecar` is already a palindrome, therefore it can be partitioned into one group.
- `fastcar` does not contain any non-trivial palindromes, so it must be partitioned as (`f`, `a`, `s`, `t`, `c`, `a`, `r`).
- `aaadbccb` can be partitioned as (`aaa`, `d`, `bccb`).

Input

Input begins with the number *n* of test cases. Each test case consists of a single line of between 1 and 100 lowercase letters, with no whitespace within.

The input must be read from standard input.

Output

For each test case, output a line containing the minimum number of groups required to partition the input into groups of palindromes.

The output must be written to standard output.

Sample Input	**Sample Output**

`3`
`racecar`
`fastcar`
`aaadbccb`
	`1`
	`7`
	`3`