

Submissions

Cheating a Boolean Tree

5pt	Not attempted 1495/1706 users correct (88%)
10pt	Not attempted 1313/1435 users correct (91%)

Triangle Areas

5pt	Not attempted 1177/1733 users correct (68%)
15pt	Not attempted 163/518 users correct (31%)

Star Wars

10pt	Not attempted 192/398 users correct (48%)
20pt	Not attempted 128/206 users correct (62%)

PermRLE

5pt	Not attempted 1322/1388 users correct (95%)
30pt	Not attempted 83/322 users correct (26%)

Top Scores

Eryx	100
bmerry	100
austrin	100
ACRush	100
darnley	100
mystic	100
Ahyangyi	100
halyavin	100
tourist	100
misof	100

Problem D. PermRLE

This contest is open for practice. You can try every problem as many times as you like, though we won't keep track of which problems you solve. Read the Quick-Start Guide to get started.

Small input
5 points

Solve D-small

Large input
30 points

Solve D-large

Problem

You've invented a slight modification of the run-length encoding (RLE) compression algorithm, called PermRLE.

To compress a string, this algorithm chooses some permutation of integers between 1 and **k**, applies this permutation to the first **k** letters of the given string, then to the next block of **k** letters, and so on. The length of the string must be divisible by **k**. After permuting all blocks, the new string is compressed using RLE, which is described later.

To apply the given permutation *p* to a block of **k** letters means to place the *p*[1]-th of these letters in the first position, then *p*[2]-th of these letters in the second position, and so on. For example, applying the permutation {3,1,4,2} to the block "abcd" yields "cadb". Applying it to the longer string "abcdefghijkl" in blocks yields "cadbgefhklij".

The permuted string is then compressed using run-length encoding. To simplify, we will consider the *compressed size* of the string to be the number of groups of consecutive equal letters. For example, the compressed size of "aabcaaaa" is 4; the first of the four groups is a group of two letters "a", then two groups "b" and "c" each containing only one letter, and finally a longer group of letters "a".

Obviously, the compressed size may depend on the chosen permutation. Since the goal of compression algorithms is to minimize the size of the compressed text, it is your job to choose the permutation that yields the smallest possible compressed size, and output that size.

Input

The first line of input gives the number of cases, **N**. **N** test cases follow.

The first line of each case will contain **k**. The second line will contain **S**, the string to be compressed.

Output

For each test case you should output one line containing "Case #**X**: **Y**" (quotes for clarity) where **X** is the number of the test case and **Y** is the minimum compressed size of **S**.

Limits

N = 20
S will contain only lowercase letters 'a' through 'z'
The length of **S** will be divisible by **k**

Small dataset

2 ≤ **k** ≤ 5
1 ≤ length of **S** ≤ 1000

Large dataset

2 ≤ **k** ≤ 16
1 ≤ length of **S** ≤ 50000

Sample

Input	Output
2	Case #1: 7
4	Case #2: 12
abcbabcbabc	
3	
abcbabcbabc	

All problem statements, input data and contest analyses are licensed under the [Creative Commons Attribution License](#).

© 2008-2017 Google [Google Home](#) - [Terms and Conditions](#) - [Privacy Policies and Principles](#)

Powered by



Google Cloud Platform