Code Jam 2015 - Round 1C

Typewriter Monkey

Problem

Your publishing house has decided to use monkeys randomly typing at keyboards to write great works of literature. You are the supervisor for one monkey with a keyboard containing **K** keys, each of which is labeled with an uppercase English letter. (There may be multiple keys displaying the same letter.) The monkey will start with an empty string and repeat the following **S** times: choose a key from its keyboard uniformly at random and press it, adding a copy of that key's letter to the right end of the string. The final resulting string will have length **S**.

You have a *target word* of length **L** that you are hoping the monkey will type. (The target word will not necessarily be a real English word.) This target word may even appear multiple times in what the monkey types. (Overlapping instances count too -- for example, if "ABA" is the target word and the monkey types "ABABA", that contains two instances of the target.)

You plan to pay the monkey one banana for each instance of the target word that it types. When you go to inspect the monkey's work, you will bring along the minimum number of bananas that you need to *ensure* that you will always have enough bananas to pay the monkey, no matter what it has typed. Then, you will pay the monkey one banana for each instance of the target word that it *actually* typed. You will keep the remaining bananas that you brought with you.

What is the expected number of bananas that you will get to keep?

Input

The first line of the input gives the number of test cases, **T**. **T** test cases follow. Each consists of three lines. The first contains three space-separated positive integers: **K**, **L**, and **S**. The second contains a string of **K** uppercase English letters representing the monkey's keyboard. The third contains a string of **L** uppercase English letters representing the target word.

Output

For each test case, output one line containing "Case #x: y", where y is the expected number of bananas you will get to keep after paying the monkey.

y will be considered correct if it is within an absolute or relative error of 10^{-6} of the correct answer. See the FAQ for an explanation of what that means, and what formats of real numbers we accept.

Limits

Memory limit: 1 GB.
1 ≤ **T** ≤ 100.

Small dataset

Time limit: 240 seconds.
1 ≤ **K** ≤ 7.
1 ≤ **L** ≤ **S** ≤ 7.

Large dataset

Time limit: 480 seconds.
1 ≤ **K** ≤ 100.
1 ≤ **L** ≤ **S** ≤ 100.

Sample

Sample Input

```
5
7 6 6
BANANAS
MONKEY
2 3 4
AA
AAA
2 1 2
AB
B
6 2 2
GOOGLE
GO
26 11 100
ABCDEFGHIJKLMNOPQRSTUVWXYZ
ROSENCRANTZ
```

Sample Output

```
Case #1: 0.0
Case #2: 0.0
Case #3: 1.0
Case #4: 0.8888889
Case #5: 9.0
```

Note that Case #5 is not within the limits for the Small dataset.

In Case #1, the monkey has no chance of typing the target word "MONKEY" even once (because his keyboard lacks most of the letters in "MONKEY"), so you do not bring any bananas along when you visit, and of course you do not pay any. Poor monkey!

In Case #2, the monkey is guaranteed to type "AAAA", which has two overlapping instances of the target word "AAA". You will bring two bananas and then pay both.

In Case #3, the monkey will produce the following outputs with equal probability (1/4 each): "AA", "AB", "BA", "BB". These have 0, 1, 1, and 2 instances of the target word, respectively. You must bring 2 bananas to be ready for the "BB" case, but you will on average pay (0 + 1 + 1 + 2) / 4 = 1.

In Case #4, the monkey has a 1/3 chance of typing a "G" first and a 1/3 chance of typing an "O" second, for a 1/9 chance of typing "GO". You will bring one banana and give it up 1/9 of the time.

In Case #5, the monkey could in theory type "ROSENCRANTZ" up to nine times, but the chances of this happening even once are so small that they are negligible compared to the acceptable margin of error for answers.