1. K Factor

The K factor of a string is defined as the number of times ‘abba’ appears as a substring.
Given two numbers N and k, **find the number of strings of length N with ‘K factor’ = k.**

- Do all calculations modulo $10^9 + 7$.
- Only lower case english alphabets are allowed.
- Overlapping matches should be counted as different substrings. For example:
  The K factor of the string ‘abbabba’ is 2.
  The K factor of the string ‘abbacdef’ is 1.

For example for N = 10 and k = 2, the number of length 10 strings with k factor 2 = 74357 strings (out of a possible 26^{10} strings).

1. The solution has to be $O(N^2)$.
2. You will get a ‘Time limit exceeded error’ if your submission takes more than 3 secs to execute.
3. You can implement in C or C++, submit a file (including main) that takes input and produces output as follows. Printing anything other than what is required by the format will make your submission fail the automatic evaluation, getting you a 0.

Input:
First line contains a single integer T, where T is the number of test cases.
The second line contains two integers N and K.
$1 <= N <= 200$
$1 <= T <= 10^5$

Output:
Output **T** numbers (Number of strings of length N, with K-factor k). Each number on a new line.

Example

Input:	**Output:**
3 4 1 7 1 10 2	1 70302 74357

2. Edit Distance using LCS

(3 SUB-PARTS)

PART A

Compute the case-insensitive minimum edit distance using the LCS algorithm.
(Do all calculations modulo $10^9 + 7$)

Input:
First line contains a single integer T, where T is the number of test cases.
The next T lines contain two strings.
$1 <= T <= 10^5$
$0 <= len(strings) <= 10^3$

Output:
Output **T** numbers (the minimum edit distance between each pair of strings, using LCS).
Each number on a new line.

Example:

Input:	**Output:**
3 ABC bD space ship ship dock	3 5 8

Explanation:
In the first test-case, ABC and bD, the operations could be

1. Delete A
2. Insert D
3. Delete C

Hence, the output is 3. (The **minimum** number of operations required, should be the output)

Note:
Please note the following.

1. *Replace/substitute* operation **should not** be supported
2. Only *insert* and *delete* operations are to be considered
3. Should be case-**in**sensitive (*'A'* is equivalent to *'a'*)
4. Do all calculations modulo $10^9 + 7$

PART B

Generally, the LCS algorithm handles only insertion and deletion operations, to compute the edit distance. Tweak the previous solution (part A) to accommodate substitution/replace as an operation.
(Do all calculations modulo $10^9 + 7$)

Input:
First line contains a single integer T, where T is the number of test cases.
The next T lines contain two strings.
$1 <= T <= 10^5$
$1 <= len(strings) <= 10^3$

Output:
Output **T** numbers (the minimum edit distance between each pair of strings, using LCS). Each number on a new line.

Example:

Input:	**Output:**
3 ABC bD space ship ship dock	2 4 4

Explanation:
In the first test-case, ABC and bD, the operations could be

1. Delete A
2. Replace C

Hence, the output is 2. (The **minimum** number of operations required, should be the output)

Note:
Please note the following.

1. *Insert*, *delete* and *replace* operations are to be considered
2. Should be case-**in**sensitive (*'A'* is equivalent to *'a'*)
3. Do all calculations modulo $10^9 + 7$

PART C

Compute the minimum weighted edit distance using the LCS algorithm where the weights are the rank of the letters in the alphabet for the insert and delete operation (A=1, B=2, Z=26, etc) and the weight of the replace operation is the distance between the letters (the weight associated with replacing A with B is 1, Z->A is 25, D->C is 1, C->D is 1, etc).
Tweak the previous solution (part B) to accommodate the weights of each operation.
(Do all calculations modulo $10^9 + 7$)

Input:
First line contains a single integer T, where T is the number of test cases.
The next T lines contain two strings.
$1 <= T <= 10^5$
$1 <= len(strings) <= 10^3$

Output:
Output **T** numbers (the minimum edit distance between each pair of strings, using LCS). Each number on a new line.

Example:

Input:	**Output:**
2 ABC bD ship dock	2 27

Explanation:
In the first test-case, ABC and bD, the operations could be

1. Delete A => 1
2. Replace C with D => 1

Hence, the output is 2. (The **minimum weighted sum** of operations required, should be the output)

Note:
Please note the following.

1. *Insert*, *delete* and *replace* operations are to be considered
2. Should be case-**in**sensitive (*'A'* is equivalent to *'a'*)
3. Do all calculations modulo $10^9 + 7$