# 2333 Enigma

During the Second World War, the German military forces mainly used one special machine to secure their communication: the *Enigma* (see Figure 4). Breaking the Enigma cipher is one of the main success stories of Allied cryptanalysis and the triumph was mainly attributed to the emergence of digital computation and the genius of the people working at Bletchley Park, the secret cryptanalysis headquarters in England. The reason for this is that, while Enigma is certainly secure against pen and paper attacks, it is quite easily breakable using digital computers.

The Enigma was a rotor machine, a cipher method which was popular at that time. A rotor is an insulated disk on which electrical contacts, one for each letter of the alphabet, are placed uniformly around the periphery and on each side. An internal conduction path through the insulating material connects contacts in pairs, one on each side of the disk. An electric current entering on one side travels on an internal path through the rotor cross-section, emerging at one of the contacts on the other side (see Figure 5 for a 3D visualisation of two rotors). Figure 6 shows a schematic side view of the complete rotor system. It shows that the Enigma has three rotors $\pi_0$, $\pi_1$ and $\pi_2$ plus an additional reflecting rotor $\pi_R$.

Figure 4: An Enigma machine (picture source: http://www.nsa.gov/museum/enigma.html).

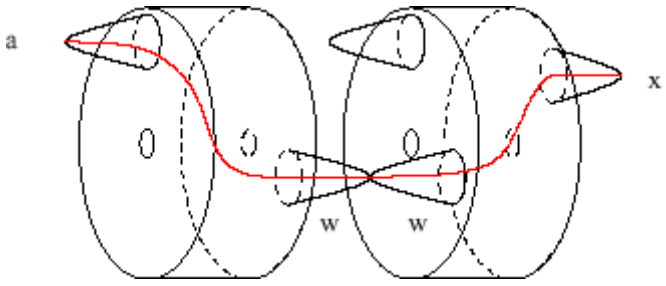


Figure 5: 3D view of two rotors.

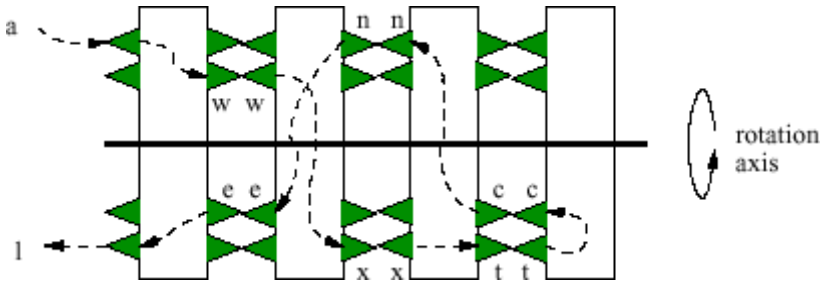


Figure 6: Side view of the Enigma's rotor system.

The input to Enigma is a stream of alphabetic characters without blanks. Every character is subject to the following steps:

1. The plaintext is subject to an initial permutation $IP$ which is implemented by a *plugboard*.
2. The character resulting from step 1 is sent through the three rotors $\pi_0$, $\pi_1$ and $\pi_2$.
3. The resulting character is then sent through the reflecting rotor $\pi_R$.
4. The character from step 3 is passed back through the rotors $\pi_2$, $\pi_1$ and $\pi_0$ (i.e., in the opposite direction).
5. The character from step 4 is subject to the inverse $IP^{-1}$ of the initial permutation $IP$.

The interesting point about the use of rotors is that after processing each character, every rotor might be rotated by a certain angle (i.e., a certain amount of letters) before processing the next character. With the Enigma, rotor $\pi_0$ is rotated by one in anti-clockwise direction with every new character. When $\pi_0$ has finished one round (i.e., after processing 26 characters), rotor $\pi_1$ moves by one character. Similarly, rotor $\pi_2$ is rotated by one character when $\pi_1$ has finished one revolution, and the reflecting rotor $\pi_R$ moves when $\pi_2$ has finished its rotation. Obviously, $\pi_R$ is the slowest of the four rotors.

The process described above can be used both for encryption and decryption, provided that the permutation $\pi_R$ implemented by the reflecting rotor is an involution. That means $\pi_R = \pi_R^{-1}$ , or, equivalently, $\xi = \pi_R(\zeta)$ whenever $\zeta = \pi_R(\xi)$. You may assume that this condition holds.

The secret key of the Enigma consists of (1) the rotors $\pi_0$, $\pi_1$, $\pi_2$ and $\pi_R$, (2) the plugboard permutation $IP$ , and (3) the initial rotational displacements $k_0, k_1, k_2, k_R$ of $\pi_0$, $\pi_1$, $\pi_2$ and $\pi_R$ (see below). The rotors were changed infrequently and were selected from a set of four possible rotors in the Wehrmacht model.

You are time-warped to Bletchley Park together with your laptop and should help to decipher some messages which have been intercepted over the day. You are given the entire ciphertext, parts of the plaintext, and parts of the Enigma key. Your task is to determine the correct key and finally complete the plaintext by decoding the ciphertext.

## Input

The first line contains the number of scenarios.

Each scenario begins with the secret key of the Enigma. The secret key is specified by 6 lines. The first four lines contain a specification of the rotors $\pi_0$, $\pi_1$, $\pi_2$ and $\pi_R$ as a sequence of lowercase alphabetic characters. Character $i$ ($1 \leq i \leq 26$) gives the mapping of the $i$-th character of the alphabet (e.g., '`bha...`' means that '`a`' is mapped to '`b`', '`b`' is mapped to '`h`', '`c`' is mapped to '`a`' etc.). Physically, the sequence of characters is given in clockwise direction looking from the front of the rotor stack $\pi_0, \ldots, \pi_R$.

After the rotors follows a similar line giving the plugboard permutation $IP$. Finally, the sixth line of the key gives the initial displacement $k_0, k_1, k_2, k_R$ of the four rotors $\pi_0$, $\pi_1$, $\pi_2$ and $\pi_R$ as a string of four characters where '`a`' means that the rotor is in its original position (as defined by the rotor specification above), '`b`' means that it is rotated by one position in the usual way etc. For example, '`dgaa`' means that rotor $\pi_0$ has initial displacement 3, $\pi_1$ has 6, and $\pi_2$, $\pi_R$ are both in their original position.

After the key follow two lines, each containing at least 1 and at most 80 lowercase letters, and no other characters. The first line contains the plaintext while the second line contains the ciphertext.

The plaintext and any part of the key may be *incomplete*, i.e., some positions in the strings may be question marks '`?`'. The number of question marks in the input will be at most 3.

## Output

The output for each scenario begins with a line containing '`Scenario #`$i$`:`', where $i$ is the number of the scenario starting at 1. In the next line you are to output the completed, decrypted plaintext. You can assume that a solution exists and that it is unique. Terminate the output for each scenario with a blank line.

## Sample Input

```
2
wfbtiznuvcqejpokshxgmadyrl
hmrgnqpkjcaivwluebfzsyxtdo
druahlbfzvgmwckxpiqysontje
owtvskypjifmluahrqecndbzgx
?bcdefghijklmnopqrstuvwxyz
aaaa
manyorganizationsrelyoncom??ters
grsuztldsznkwnerdpfbovvqnobkyiqn
oqzunvhtxwryfebicmjpklsgda
zupogrskynxtwdfqvbliejcmha
kzvlyjuodmscewxtfbphriqgna
gbcnylaztwkfmdspqvoiurjxeh
rfyhkxbuvplgtqmdiewjosznca
dmeo
???
ave
```

## Sample Output

```
Scenario #1:
manyorganizationsrelyoncomputers

Scenario #2:
acm
```