Vote Verifier

Computer Science Society
Programming Contest
Fall 2008

Having just completed a national election with the largest voter turnout in decades, one wonders how well the votes were actually counted. I've voted many times in my life, but have never been able to verify that my vote was correctly counted, or that the vote totals reported were accurate. Computer scientists have employed cryptographic techniques to build end-to-end verifiable voting systems. They allow you to verify your vote, allow everyone to verify the vote totals, and allow nobody other than you to determine how you voted. In this problem, we'll examine a simplified voting system that contains a few (but not all) of the basic features of end-to-end verifiable voting systems.

A list of n candidates $(1, 2, 3, \ldots, n)$ is running for one seat. Separate (plaintext) ballots list the candidates in potentially different orders—random permutations of the list $1, 2, 3, \ldots, n$. Voters are given one of the ballots and cast their vote by marking the index at which their candidate appears on their ballot permutation. The ballot is published by the registrar with the voters mark on it, but with the ballot permutation encrypted using a key given only to the voter. The voter can use the key to verify that their ballot was correctly recorded, but nobody else can determine who they voted for. In this problem, we are given published ballots and corresponding keys and verify which candidates received the votes.

Input Format

The first line of input contains a positive integer $n \leq 9$, representing the number of candidates running for one seat. The remaining lines of input representing individual votes containing an *index* mark, a *ciphertext ballot* and a *key* separated by white space. The index is a single digit between 1 and n, inclusive. The ciphertext ballot and key are n-digit numbers whose individual digits are between 0 and 9, inclusive. The ciphertext ballot and the key can be "added" (denoted $\oplus$) to produce an n-digit *plaintext ballot* by adding the corresponding digits of the ciphertext ballot and key modulo 10. For example $5 \oplus 6 = 1$ and $283 \oplus 139 = 312$. The digits of the resulting plaintext ballot, read from left-to-right, are the permutation of the candidate list $1, 2, 3, \ldots, n$ appearing on the original ballot.

Output Format

For each line of input representing a vote, compute and print the plaintext ballot, index and candidate receiving the vote (i.e. candidate on the plaintext ballot at the index position), as shown in the output sample.

Input Sample

```
5
3 19823 16311
3 97886 54549
3 27339 37803
2 79394 45941
4 27350 96904
```

Output Sample

```
plaintext ballot: 25134  index: 3  vote for candidate: 1
plaintext ballot: 41325  index: 3  vote for candidate: 3
plaintext ballot: 54132  index: 3  vote for candidate: 1
plaintext ballot: 14235  index: 2  vote for candidate: 4
plaintext ballot: 13254  index: 4  vote for candidate: 5
```