2172 Merkle-Hellman Knapsack Public Key Cryptosystem

In this problem you will implement the Merkle-Hellman Knapsack cryptosystem. The encryption and decryption algorithms are based on solving a knapsack problem. The knapsack problem consists of finding a way to select some of the items to be packed such that their sum (the amount of space they take up) exactly equals the knapsack capacity (the target). Formally the knapsack problems is as follows: given a set $S = \{a_1, a_2, \ldots a_n\}$ and a target sum T, where each $a_i \geq 0$, is there a selection vector $V = \{v_1, v_2, \ldots v_n\}$ each of whose elements is 0 or 1, such that $\sum_i (a_i * v_i) = T$.

The idea behind Merkle-Hellman scheme is to encode a binary message as a solution to a knapsack problem reducing the ciphertext to the target sum obtained by adding terms corresponding to 1s in the plaintext. That is, blocks of plain text are converted to knapsack sums by adding into the sum the terms that match with 1bits in the plaintext as shown below:

Plaintext:		1	0	1	0	0	1
Knapsack:		1	2	5	9	20	43
Ciphertext:	49						

Plaintext:		0	1	1	0	1	0
Knapsack:		1	2	5	9	20	43
Ciphertext:	27						

The receiver will have to use the ciphertext (the target sum) to recover the plaintext.

A knapsack problem is said to be "easy" if $\forall k \; a_k > \sum_j^{(k-1)} a_j$. The solution for the easy knapsack problem is straightforward, and can be achieved in $O(n)$. Furthermore, there exists a way for transforming an easy knapsack set S into a non-easy knapsack H simply by multiplying the elements of S by w mod n where w and n are well chosen integers. Formally the terms of H are: $h_i = (w * s_i) \bmod n$.

For example if $S = \{1, 2, 4, 9\}$, $w = 15$, and $n = 17$ then $H = \{15, 13, 9, 6\}$.

Encryption algorithm (executed by the sender)

H is called the public key and is made public by the receiver

1. Choose w and n such that $n > \max(S)$ and n is prime, and $w < n$. Construct H from S, w, and n.
2. Sender uses H to encipher blocks of of m bits $P_0 = [p_1, p_2 \ldots p_m]$, $P_1 = [p_{m+1}, p_{m+2} \ldots p_{2m}]$ and so forth (m is the number of terms in H), as follows: $T_i = P_i * H = \sum_j (p_j * h_j)$. Thus $T_0 = P_0 * H$, $T_1 = P_1 * H$ and so on. T_i are then transmitted to the receiver via a reliable channel.

Decryption algorithm (executed by the receiver)

The tuple (S, n, w) is called the private key and is kept secret by the receiver

1. Use w and n to compute w^{-1} where $w^{-1} * w = 1 \bmod n$. If n is prime then $w^{-1} = w^{(n-2)} \bmod n$
2. Compute $A_i = w^{-1} * T_i \bmod n$
3. Find P_i by solving $A_i = P_i * S$

Input

The input consists of N test cases. The number of them (N) is given on the first line of the input file. Each test case begins with a line containing a plaintext to be encrypted. The second line contains the number of elements (m) in the knapsack S that show in the third line. The knapsack elements are positive integers separated by space. The fourth line of each text case contains n and w in this order separated by space.

Output

Print exactly 3 lines for each test case. The first line should contain the encrypted values of the plaintext of the set separated by space. The second line should contain the plaintext obtained by applying the decryption algorithm to the plaintext, preceded by '`Recovered plain text: `'. The third line should contain the value of w^{-1}

Sample Input

```
2
Salaam!
4
1 2 4 9
17 15
hello there?
4
1 2 4 9
19 7
```

Sample Output

```
29 25 22 16 22 28 22 16 22 16 22 44 9 16 0 24
Recovered plain text: Salaam!
8
23 7 23 20 23 21 23 21 23 36 9 0 29 14 23 7 23 20 29 9 23 20 15 36 0 16
Recovered plain text: hello there?
11
```