Problem F. Filter

Input file: `filter.in`
Output file: `filter.out`

You are working on a new high-performance database engine — Instant Compression and Processing Codec (ICPC). ICPC stores user activity records. Each user activity record has an integer *user identifier*. The records are stored in a number of data files. Each data file is compressed and can contain records from multiple users, however ICPC has to process queries that look for a specific subsets of users. In order to do so, there has to be a way to quickly determine which data files may contain records for a specific user before attempting to decompress them, which may be a long and CPU-consuming process.

ICPC uses an algorithm called *Bloom Filter*. The way it is implemented in ICPC is described below. For each ICPC database the following integer parameters are chosen:

- m is the number of bits in the filter;
- f is the number of hash functions in the filter;
- a_i are the parameters for hash functions for $0 \le i < f$.

A value of the bloom filter is computed for each data file. The data file's bloom filter is a vector of m bits. A bit number j ($0 \le j < m$) is set to one if and only if there is a record in this data file for some user identifier u_k, such that for some hash function i ($0 \le i < f$) the following equality holds:

$$j = (u_k \cdot a_i) \bmod m \tag{1}$$

Your task is to implement ICPC filtering logic. You are given filter parameters and values for a number of data files and a set of user identifiers. Your task is determine which data files may contain record with at least one user identifier from the specified set. A data file may contain a record with a user identifier u_k if and only if for all i ($0 \le i < f$) all the bits j given by equality (1) in its filter value are set to one.

Input

The first line of the input file contains filter parameters — integer numbers m, f, and a_i for $0 \le i < f$ ($1 \le m \le 1000$, $1 \le f \le 100$, $1 \le a_i < 2^{31}$).

The second line of the input file contains an integer n — the number of data files ($1 \le n \le 1000$). Each of the following n lines contains bloom filter value of the corresponding file in hexadecimal form. Each value is represented by a string of $\lceil m/4 \rceil$ hexadecimal digits (one of `0123456789abcdef`). The first digit of the string represents bits 0–3 of the value (stored in order from the least significant bit of a hexadecimal digit to the most significant bit), the second digit — bits 4–7, the third — 8–11, etc. When $m \bmod 4 \ne 0$, then the last hexadecimal digit represents the last $m \bmod 4$ bits of the value in its least significant bits.

The following line of the input file contains an integer q — the number of user identifiers in a query ($1 \le q \le 1000$), followed by q integers u_k — the set of distinct user identifiers in the query ($1 \le u_k < 2^{31}$).

Output

Write a line with the integer number s to the output file — the number of data files that may contain a record with at least one user identifier from the specified set, followed by s numbers d_t ($0 \le d_t < n$) — the 0-based numbers of the corresponding data files in ascending order.

Sample input and output

filter.in	filter.out
23 4 3 5 7 11 3 effde7 c07902 0800c1 3 2 4 6	2 0 2