Video Service

You are the single engineer who has newly been placed in charge of all of the servers at Metflicks after the previous employee quit for 3% more total compensation at a rival streaming service. After coming to terms with the fact that you are doomed to manage the terabytes of data requested of your precious servers each day, you pull yourself up by your bootstraps and get to work. Once you finally figure out how everything works, you stumble upon a note scrawled in what looks like ketchup tucked away in your desk. It is a hastily written list titled DO NOT SERVE UNDER ANY CIRCUMSTANCES, with many IP addresses listed beneath. You find another list written in what smells like mustard titled BAD DATA, with very sketchy descriptions of packet payloads that when strung together, cause all of your servers to catch fire.

Write a program that, given a list of IP addresses and data sequences, rejects all packets that contain IP addresses that are not to be served and halts acknowledgements when multiple successive packets are combined to form bad data. Packets with lower sequence numbers contain higher significance bits.

In addition, known valid users can make bad data requests, putting them on the naughty list. Once a user has been marked as DO NOT SERVE, they cannot get off this list and are permanently barred from using Metflicks.

Input

The first line of input is a number $2 \leq a \leq 2^{20}$ of DO NOT SERVE IP addresses. The next a lines will contain 32-bit representations of IP addresses

The next line of input is a number $2 \leq b \leq 2^{11}$ of possible bitstrings of bad data. The next b lines will contain bitstrings of length 2^5.

The following line of input is a number $2^2 \leq p \leq 2^{500}$, where you will receive p total packets. The next p lines will be packets in the form:

32-bit source IP address i	32-bit current sequence number c	32-bit total sequence number t	32-bit data bitstring d

The remaining input is a number $2 \leq u \leq 2^{20}$ representing the number of user IP address to test. The following u lines of input are IP addresses that may or may not have sent a request to your servers.

Output

For each IP address u, print 1 to mark that this user only had good requests or print 0 to mark that this user either was originally on the DO NOT SERVE list or earned their way on by sending bad data. A user moves from being good to bad if they send three bad messages in the same chain. Print nothing if this user did not make a request.

Languages

This functions in java, python and C++, or pretty much any language that has support of integers over size 2^{500}. Runtime should not be a problem unless I/O is extremely slow.

Sample Input

```
3
01111000000010011110111111010011
01001010010111110111101110110111
00111111100100000000100001111100
4
00011110010011011111101110001011
00011000101001010000111001110110
00000011010101100000011110100011
01010000100100001110001011110110
10
01001111011000011110011011000010000000000000000000000000000000001000000000000000000000000000001000000110001010010100000110001110110
01001111011000011110011011000010000000000000000000000000000000001000000000000000000000000000001000001111000110010011110110000010001
01001111011000011110011011000010000000000000000000000000000000001000000000000000000000000000001000011011100100110100000101010011000
01001111011000011110011011000010000000000000000000000000000000001000000000000000000000000000001000001111001001101111110110001011
01001111011000011110011011000010000000000000000000000000000000001010000000000000000000000000001000010001010101100111111101110011000
01001111011000011110011011000010000000000000000000000000000000001000000000000000000000000000001000000000110101011011111100000
01001111011000011110011011000010000000000000000000000000000000001110000000000000000000000000001000001010010010111111010101111101111
01001111011000011110011011000010000000000000000000000000000000001000000000000000000000000000001000000110001010010100000110001110110
01001101011110110110111111001000000000000000000000000000000000001000000000000000000000000000001001010000100100001110001011110110
01001101011110110110111111001000000000000000000000000000000000001000000000000000000000000000001000011000101001010000110001110110
6
01001111011000011110011011000010
01001101011111011101101111111001
00111111100100000000100001111100
00101001101001111111011011101000
01001010010111110111101110110111
00100111001101001011100011110111
```

Sample Output

0100