Kekocity's population consist of **N** gnomes numbered with unique ids from **1** to **N**. As they are very joyful gnomes, they usually send jokes to their friends right after they get any (even if they knew it before) via their social network named as Mybeard. Mybeard became popular in the city because of message auto-deletion. It takes exactly one minute to read and resend joke to mates.

Mayor of Kekocity, Mr. Shaikhinidin, is interested in understanding how the jokes are spread. He gives you database of Mybeard social network, and wants you to answer some queries on it.

You will be given a list of friends for every gnome and **M** queries of the following type: Who will receive a message with joke after exactly **K** minutes, if the creator of joke was gnome with id **x**?

Input

The first line contains a single integer **N** denoting the number of gnomes.

The next **N** lines contain the the matrix **g[N][N]**. Each of the **i**-th line, will contain **N** space separated integers - **j**-th of those will denote **g[i][j]**. If gnome **j** is friend of gnome **i**, then **g[i][j]** is 1. Otherwise it will be zero. Plese note that the friendship relationship is not bidirectional, i.e. it might happen that g[i][j] may not be equal to g[j][i]. Also one can be friend of itself also, i.e. g[i][i] may be equal to 1.

The next line contains a single integer **M** denoting the number of queries. The next **M** lines contain two integers **k** and **x** described above.

Output

For each query, output two lines.

In the first line, output how many gnomes will know the joke after **k** minutes.

In the second line, print these ids (numbering) of these gnomes in increasing order. If no one will know the joke after **K** minutes, then print -1 in this line.

Constraints

- **$1 \le N \le 500$**
- **$1 \le M \le 500$**
- **$0 \le k \le 10^9$**
- **$1 \le x \le N$**
- **$0 \le g[i][j] \le 1$**

Sample Input

5

0 1 0 0 0

0 0 1 1 0

1 0 0 0 0

0 0 0 1 0

0 0 0 0 0

4

3 1

10000 1

0 5

1 5

Sample Output

2

1 4

2

2 4

1

5

0

-1

```
import java.util.*;

public class Program {
	public static void main(String args[]) {
```

```
		Scanner input = new Scanner(System.in);
		int n = input.nextInt();
		long g[][][] = new long[30][500][9];
		for (int i = 0; i < n; i++)
			for (int j = 0; j < n; j++)
				if (input.nextInt() == 1)
					g[0][i][j / 60] |= 1l << (j % 60);
		for (int t = 1; t < 30; t++)
			for (int i = 0; i < n; i++)
				for (int j = 0; j < n; j++)
					if ((g[t - 1][i][j / 60] & (1l << (j % 60))) != 0)
						for (int k = 0; k < 9; k++)
							g[t][i][k] |= g[t - 1][j][k];
		int m = input.nextInt();
		while ((m--) > 0) {
			int len = input.nextInt(), x = input.nextInt() - 1;
			long mask[] = new long[9];
			mask[x / 60] = 1l << (x % 60);
			for (int t = 0; t < 30; t++)
				if ((len & (1l << t)) != 0) {
					long newmask[] = new long[9];
					for (int i = 0; i < n; i++)
						if ((mask[i / 60] & (1l << (i % 60))) != 0)
							for (int j = 0; j < 9; j++)
								newmask[j] |= g[t][i][j];
					for (int i = 0; i < 9; i++)
						mask[i] = newmask[i];
				}
			int ans[] = new int[n];
			int cnt = 0;
			for (int i = 0; i < n; i++)
				if ((mask[i / 60] & (1l << (i % 60))) != 0)
					ans[cnt++] = i + 1;
			System.out.println(cnt);
			for (int i = 0; i < cnt; i++)
				System.out.print(ans[i] + (i == cnt - 1 ? "\n" : " "));
			if (cnt == 0)
				System.out.println(-1);
		}
	}
}
```