H. Ultrastar

```
   Time limit:  1 seconds
Memory limit:  65535 kBytes
```

Description

John Doe is planning a party again, but this time it will be a karaoke party! To make things more fun, he wants to use UltraStar, an open-source karaoke game which scores the singers automatically.

John downloads n songs from different places on the internet, he loads all of them into the game, starts up the software, and... surprise, it hangs! It looks like there are k bad songs among the n, and if John loads even one of them into the game, it won't start. He only has t minutes until the party and it takes exactly one minute to start up the game with a certain subset of the n songs. Help John in making the party a success, by determining all of the bad songs!

Input and output

This is an *interactive problem*, which means that you should not read the input and write the output in the usual way, rather you have to interact with the grader program by following these steps:

1. Read from the standard input the values of n, k, and t.
2. In order to start up the game with a certain subset of songs, write to the standard output a line having the format `1` m $i_1, i_2, \ldots, i_m$ (with all numbers separated by a single space), where m is the number of songs in the subset and $i_1, \ldots, i_m$ are the indices of the songs in any order. It's important to flush the output buffer after this (eg. by using `endl` in C++ or `flush` in Pascal).
3. After each query you should read a single number from the standard input, then proceed to step 2 or step 4. The number will be 0 if the game doesn't start up, and 1 if it does.
4. In order to give a final answer write to the standard output a line having the format `2` $j_1, j_2, \ldots, j_k$ (with all numbers separated by a single space), where $j_1, \ldots j_k$ are the indices of the bad songs.

Constraints

- $1 \leq n \leq 1000$
- $1 \leq k \leq n$
- $0 \leq t \leq 100\ 000$

- $1 \leq m \leq n$ otherwise you will receive a *Wrong Answer* judgement.
- $i_1, \ldots, i_m$ should be all distinct, otherwise you will receive a *Wrong Answer* judgement.
- $j_1, \ldots, j_k$ should be all distinct, otherwise you will receive a *Wrong Answer* judgement.
- If you start up the game more than t times, the grader will stop the execution of your program and you will receive a *Wrong Answer* judgement.
- The songs are numbered from 1 to n, if you output an index outside of the $[1, n]$ interval you will receive a *Wrong Answer* judgement.
- Your solution will be evaluated on 20 test cases having the following values:

Case	n	k	t
1	4	2	1
2	5	2	5
3	6	3	5
4	1	1	0
5	1000	20	226
6	3	3	0
7	900	30	289
8	998	499	1
9	999	499	1
10	997	499	1004
11	999	500	999
12	900	300	1448
13	800	100	647
14	1000	250	2
15	999	998	1747
16	1000	200	1137
17	1000	908	1978
18	1000	100	736
19	1000	10	36
20	100	40	98

- *For each test case* it is guaranteed that there is a deterministic (non-randomized) strategy that solves the problem using no more than t queries. *Hint:* You may need to use a different strategy for ***odd*** and **even** cases, respectively.

Example

If $n = 4, k = 2$, and the bad songs are the ones with indices 3 and 4, a possible interaction:

Standard output	Standard Input
	4 2 1
1 2 1 2	
	1
2 3 4	