Codeforces β
Sponsored by Telegram

Testing Round #13

A. Santa Claus and Candies

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Santa Claus has n candies, he dreams to give them as gifts to children.

What is the maximal number of children for whose he can give candies if Santa Claus want each kid should get *distinct* positive integer number of candies. Santa Class wants to give all n candies he has.

Input

The only line contains positive integer number n $(1 \le n \le 1000)$ — number of candies Santa Claus has.

Output

Print to the first line integer number k — maximal number of kids which can get candies.

Print to the second line k *distinct* integer numbers: number of candies for each of k kid. The sum of k printed numbers should be exactly n.

If there are many solutions, print any of them.

Examples

input
```
5
```
output
```
2
2 3
```

input
```
9
```
output
```
3
3 5 1
```

input
```
2
```
output
```
1
2
```

B. Interactive Bulls and Cows (Easy)

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

This problem is a little bit unusual. Here you are to implement an interaction with a testing system. That means that you can make queries and get responses in the online mode. Please be sure to use the stream flushing operation after each query's output in order not to leave part of your output in some buffer. For example, in C++ you've got to use the `fflush(stdout)` function, in Java — call `System.out.flush()`, and in Pascal — `flush(output)`.

Bulls and Cows (also known as Cows and Bulls or Pigs and Bulls or Bulls and Cleots) is an old code-breaking paper and pencil game for two players, predating the similar commercially marketed board game Mastermind.

On a sheet of paper, the first player thinks a secret string. This string consists only of digits and has the length 4. The digits in the string **must** be all different, no two or more equal digits are allowed.

Then the second player tries to guess his opponent's string. For every guess the first player gives the number of matches. If the matching digits are on their right positions, they are "bulls", if on different positions, they are "cows". Thus a response is a pair of numbers — the number of "bulls" and the number of "cows". A try can contain equal digits.

More formally, let's the secret string is s and the second player are trying to guess it with a string x. The number of "bulls" is a number of such positions i $(1 \le i \le 4)$ where $s[i] = x[i]$. The number of "cows" is a number of such digits c that s contains c in the position i (i.e. $s[i] = c$), x contains c, but $x[i] \ne c$.

For example, the secret string is "`0427`", the opponent's try is "`0724`", then the answer is 2 bulls and 2 cows (the bulls are "0" and "2", the cows are "4" and "7"). If the secret string is "`0123`", the opponent's try is "`0330`", then the answer is 1 bull and 1 cow.

In this problem you are to guess the string s that the system has chosen. You only know that the chosen string consists of 4 distinct digits.

You can make queries to the testing system, each query is the output of a single 4-digit string. The answer to the query is the number of bulls and number of cows. If the system's response equals "4 0", that means the interaction with your problem is over and the program must terminate. That is possible for two reasons — the program either guessed the number x or made an invalid action (for example, printed letters instead of digits).

Your program is allowed to do at most 50 queries.

You can hack solutions of other participants providing a 4-digit string containing distinct digits — the secret string.

Input

To read answers to the queries, the program must use the standard input.

The program will receive pairs of non-negative integers in the input, one pair per line. The first number in a pair is a number of bulls and the second one is a number of cows of the string s and the string x_i printed by your program. If the system response equals "4 0", then your solution should terminate.

The testing system will let your program read the i-th pair of integers from the input only after your program displays the corresponding system query in the output: prints value x_i in a single line and executes operation `flush`.

Output

The program must use the standard output to print queries.

Your program must output requests — 4-digit strings $x_1, x_2, \ldots$, one per line. After the output of each line the program must execute `flush` operation. The program should read the answer to the query from the standard input.

Your program is allowed to do at most 50 queries.

Examples

input

```
0 1
2 0
1 1
0 4
2 1
4 0
```

output

```
8000
0179
3159
3210
0112
0123
```

Note

The secret string s in the example is "`0123`".

C. Interactive Bulls and Cows (Hard)

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

The only difference from the previous problem is the constraint on the number of requests. In this problem your program should guess the answer doing at most 7 requests.

This problem is a little bit unusual. Here you are to implement an interaction with a testing system. That means that you can make queries and get responses in the online mode. Please be sure to use the stream flushing operation after each query's output in order not to leave part of your output in some buffer. For example, in C++ you've got to use the `fflush(stdout)` function, in Java — call `System.out.flush()`, and in Pascal — `flush(output)`.

Bulls and Cows (also known as Cows and Bulls or Pigs and Bulls or Bulls and Cleots) is an old code-breaking paper and pencil game for two players, predating the similar commercially marketed board game Mastermind.

On a sheet of paper, the first player thinks a secret string. This string consists only of digits and has the length 4. The digits in the string **must** be all different, no two or more equal digits are allowed.

Then the second player tries to guess his opponent's string. For every guess the first player gives the number of matches. If the matching digits are on their right positions, they are "bulls", if on different positions, they are "cows". Thus a response is a pair of numbers — the number of "bulls" and the number of "cows". A try can contain equal digits.

More formally, let's the secret string is s and the second player are trying to guess it with a string x. The number of "bulls" is a number of such positions i ($1 \le i \le 4$) where $s[i] = x[i]$. The number of "cows" is a number of such digits c that s contains c in the position i (i.e. $s[i] = c$), x contains c, but $x[i] \ne c$.

For example, the secret string is "`0427`", the opponent's try is "`0724`", then the answer is 2 bulls and 2 cows (the bulls are "0" and "2", the cows are "4" and "7"). If the secret string is "`0123`", the opponent's try is "`0330`", then the answer is 1 bull and 1 cow.

In this problem you are to guess the string s that the system has chosen. You only know that the chosen string consists of 4 distinct digits.

You can make queries to the testing system, each query is the output of a single 4-digit string. The answer to the query is the number of bulls and number of cows. If the system's response equals "4 0", that means the interaction with your problem is over and the program must terminate. That is possible for two reasons — the program either guessed the number x or made an invalid action (for example, printed letters instead of digits).

Your program is allowed to do at most 7 queries.

You can hack solutions of other participants providing a 4-digit string containing distinct digits — the secret string.

Input

To read answers to the queries, the program must use the standard input.

The program will receive pairs of non-negative integers in the input, one pair per line. The first number in a pair is a number of bulls and the second one is a number of cows of the string s and the string x_i printed by your program. If the system response equals "4 0", then your solution should terminate.

The testing system will let your program read the i-th pair of integers from the input only after your program displays the corresponding system query in the output: prints value x_i in a single line and executes operation `flush`.

Output

The program must use the standard output to print queries.

Your program must output requests — 4-digit strings $x_1, x_2, \ldots$, one per line. After the output of each line the program must execute `flush` operation. The program should read the answer to the query from the standard input.

Your program is allowed to do at most 7 queries.

Examples

input

```
0 1
2 0
1 1
0 4
2 1
4 0
```

output

```
8000
0179
3159
3210
0112
0123
```

Note

The secret string s in the example is "`0123`".