Codeforces Beta Round 4 (Div. 2 Only)

A. Watermelon

1 second, 64 megabytes

One hot summer day Pete and his friend Billy decided to buy a watermelon. They chose the biggest and the ripest one, in their opinion. After that the watermelon was weighed, and the scales showed w kilos. They rushed home, dying of thirst, and decided to divide the berry, however they faced a hard problem.

Pete and Billy are great fans of even numbers, that's why they want to divide the watermelon in such a way that each of the two parts weighs even number of kilos, at the same time it is not obligatory that the parts are equal. The boys are extremely tired and want to start their meal as soon as possible, that's why you should help them and find out, if they can divide the watermelon in the way they want. For sure, each of them should get a part of positive weight.

Input

The first (and the only) input line contains integer number w ($1 \le w \le 100$) — the weight of the watermelon bought by the boys.

Output

Print `YES`, if the boys can divide the watermelon into two parts, each of them weighing even number of kilos; and `NO` in the opposite case.

input
8
output
YES

For example, the boys can divide the watermelon into two parts of 2 and 6 kilos respectively (another variant — two parts of 4 and 4 kilos).

B. Before an Exam

0.5 second, 64 megabytes

Tomorrow Peter has a Biology exam. He does not like this subject much, but d days ago he learnt that he would have to take this exam. Peter's strict parents made him prepare for the exam immediately, for this purpose he has to study not less than $minTime_i$ and not more than $maxTime_i$ hours per each i-th day. Moreover, they warned Peter that a day before the exam they would check how he has followed their instructions.

So, today is the day when Peter's parents ask him to show the timetable of his preparatory studies. But the boy has counted only the sum of hours $sumTime$ spent him on preparation, and now he wants to know if he can show his parents a timetable $schedule$ with d numbers, where each number $schedule_i$ stands for the time in hours spent by Peter each i-th day on biology studies, and satisfying the limitations imposed by his parents, and at the same time the sum total of all $schedule_i$ should equal to $sumTime$.

Input

The first input line contains two integer numbers d, $sumTime$ ($1 \le d \le 30$, $0 \le sumTime \le 240$) — the amount of days, during which Peter studied, and the total amount of hours, spent on preparation. Each of the following d lines contains two integer numbers $minTime_i$, $maxTime_i$ ($0 \le minTime_i \le maxTime_i \le 8$), separated by a space — minimum and maximum amount of hours that Peter could spent in the i-th day.

Output

In the first line print `YES`, and in the second line print d numbers (separated by a space), each of the numbers — amount of hours, spent by Peter on preparation in the corresponding day, if he followed his parents' instructions; or print `NO` in the unique line. If there are many solutions, print any of them.

input
1 48 5 7
output
NO

input
2 5 0 1 3 5
output
YES 1 4

C. Registration system

5 seconds, 64 megabytes

A new e-mail service "Berlandesk" is going to be opened in Berland in the near future. The site administration wants to launch their project as soon as possible, that's why they ask you to help. You're suggested to implement the prototype of site registration system. The system should work on the following principle.

Each time a new user wants to register, he sends to the system a request with his `name`. If such a `name` does not exist in the system database, it is inserted into the database, and the user gets the response `OK`, confirming the successful registration. If the `name` already exists in the system database, the system makes up a new user name, sends it to the user as a prompt and *also inserts the prompt into the database*. The new name is formed by the following rule. Numbers, starting with 1, are appended one after another to `name` (`name1`, `name2`, ...), among these numbers the least i is found so that `name`i does not yet exist in the database.

Input

The first line contains number n ($1 \le n \le 10^5$). The following n lines contain the requests to the system. Each request is a non-empty line, and consists of not more than 32 characters, which are all lowercase Latin letters.

Output

Print n lines, which are system responses to the requests: `OK` in case of successful registration, or a prompt with a new name, if the requested name is already taken.

input
4 abacaba acaba abacaba acab
output
OK OK abacaba1 OK

input

```
6
first
first
second
second
third
third
```

output

```
OK
first1
OK
second1
OK
third1
```

D. Mysterious Present

1 second, 64 megabytes

Peter decided to wish happy birthday to his friend from Australia and send him a card. To make his present more mysterious, he decided to make a *chain*. Chain here is such a sequence of envelopes $A = \{a_1, a_2, \ldots, a_n\}$, where the width and the height of the i-th envelope is strictly higher than the width and the height of the $(i - 1)$-th envelope respectively. Chain size is the number of envelopes in the chain.

Peter wants to make the chain of the maximum size from the envelopes he has, the chain should be such, that he'll be able to put a card into it. The card fits into the chain if its width and height is lower than the width and the height of the smallest envelope in the chain respectively. It's forbidden to turn the card and the envelopes.

Peter has very many envelopes and very little time, this hard task is entrusted to you.

Input

The first line contains integers n, w, h ($1 \le n \le 5000$, $1 \le w, h \le 10^6$) — amount of envelopes Peter has, the card width and height respectively. Then there follow n lines, each of them contains two integer numbers w_i and h_i — width and height of the i-th envelope ($1 \le w_i, h_i \le 10^6$).

Output

In the first line print the maximum chain size. In the second line print the numbers of the envelopes (separated by space), forming the required chain, starting with the number of the smallest envelope. Remember, please, that the card should fit into the smallest envelope. If the chain of maximum size is not unique, print any of the answers.

If the card does not fit into any of the envelopes, print number 0 in the single line.

input

```
2 1 1
2 2
2 2
```

output

```
1
1
```

input

```
3 3 3
5 4
12 11
9 8
```

output

```
3
1 3 2
```