A. Restoring Password

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

Igor K. always used to trust his favorite Kashpirovsky Antivirus. That is why he didn't hesitate to download the link one of his groupmates sent him via QIP Infinium. The link was said to contain "some real funny stuff about swine influenza". The antivirus had no objections and Igor K. run the flash application he had downloaded. Immediately his QIP Infinium said: "invalid login/password".

Igor K. entered the ISQ from his additional account and looked at the info of his main one. His name and surname changed to "H1N1" and "Infected" correspondingly, and the "Additional Information" field contained a strange-looking binary code 80 characters in length, consisting of zeroes and ones. "I've been hacked" — thought Igor K. and run the Internet Exploiter browser to quickly type his favourite search engine's address.

Soon he learned that it really was a virus that changed ISQ users' passwords. Fortunately, he soon found out that the binary code was actually the encrypted password where each group of 10 characters stood for one decimal digit. Accordingly, the original password consisted of 8 decimal digits.

Help Igor K. restore his ISQ account by the encrypted password and encryption specification.

Input

The input data contains 11 lines. The first line represents the binary code 80 characters in length. That is the code written in Igor K.'s ISQ account's info. Next 10 lines contain pairwise distinct binary codes 10 characters in length, corresponding to numbers 0, 1, ..., 9.

Output

Print one line containing 8 characters — The password to Igor K.'s ISQ account. It is guaranteed that the solution exists.

Sample test(s)

input

```
01001100100101100000010110001001011001000101100110010110100001011010100101101100
0100110000
0100110010
0101100000
0101100010
0101100100
0101100110
0101101000
0101101010
0101101100
0101101110
```

output

```
12345678
```

input

```
10101101111001000010100100011010101101110010110111011000100011011110010110001000
1001000010
1101111001
1001000110
1010110111
0010110111
1101001101
1011000001
1110010101
1011011000
0110001000
```

output

```
30234919
```

B. Friends

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

One day Igor K. stopped programming and took up math. One late autumn evening he was sitting at a table reading a book and thinking about something.

The following statement caught his attention: "Among any six people there are either three pairwise acquainted people or three pairwise unacquainted people"

Igor just couldn't get why the required minimum is 6 people. "Well, that's the same for five people, too!" — he kept on repeating in his mind. — "Let's take, say, Max, Ilya, Vova — here, they all know each other! And now let's add Dima and Oleg to Vova — none of them is acquainted with each other! Now, that math is just rubbish!"

Igor K. took 5 friends of his and wrote down who of them is friends with whom. Now he wants to check whether it is true for the five people that among them there are either three pairwise acquainted or three pairwise not acquainted people.

Input

The first line contains an integer m $(0 \le m \le 10)$, which is the number of relations of acquaintances among the five friends of Igor's.

Each of the following m lines contains two integers a_i and b_i $(1 \le a_i, b_i \le 5; a_i \ne b_i)$, where (a_i, b_i) is a pair of acquainted people. It is guaranteed that each pair of the acquaintances is described exactly once. The acquaintance relation is symmetrical, i.e. if x is acquainted with y, then y is also acquainted with x.

Output

Print "`FAIL`", if among those five people there are no either three pairwise acquainted or three pairwise unacquainted people. Otherwise print "`WIN`".

Sample test(s)

input

```
4
1 3
2 3
1 4
5 3
```

output

```
WIN
```

input

```
5
1 2
2 3
3 4
4 5
5 1
```

output

```
FAIL
```

C. Frames

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

Throughout Igor K.'s life he has had many situations worthy of attention. We remember the story with the virus, the story of his mathematical career and of course, his famous programming achievements. However, one does not always adopt new hobbies, one can quit something as well.

This time Igor K. got disappointed in one of his hobbies: editing and voicing videos. Moreover, he got disappointed in it so much, that he decided to destroy his secret archive for good.

Igor K. use Pindows XR operation system which represents files and folders by small icons. At that, m icons can fit in a horizontal row in any window.

Igor K.'s computer contains n folders in the D: disk's root catalog. The folders are numbered from 1 to n in the order from the left to the right and from top to bottom (see the images). At that the folders with secret videos have numbers from a to b inclusive. Igor K. wants to delete them forever, at that making as few frame selections as possible, and then pressing Shift+Delete exactly once. What is the minimum number of times Igor K. will have to select the folder in order to select folders from a to b and only them? Let us note that if some selected folder is selected repeatedly, then it is deselected. Each selection possesses the shape of some rectangle with sides parallel to the screen's borders.

Input

The only line contains four integers n, m, a, b ($1 \le n, m \le 10^9$, $1 \le a \le b \le n$). They are the number of folders in Igor K.'s computer, the width of a window and the numbers of the first and the last folders that need to be deleted.

Output

Print a single number: the least possible number of times Igor K. will have to select the folders using frames to select only the folders with numbers from a to b.

Sample test(s)

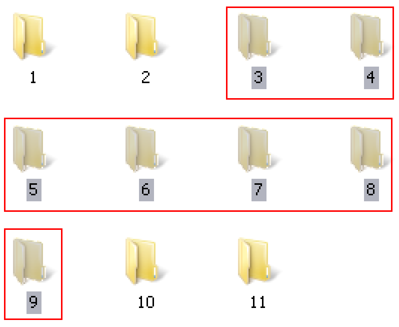
```
input
11 4 3 9
output
3
```

```
input
20 5 2 20
output
2
```

Note

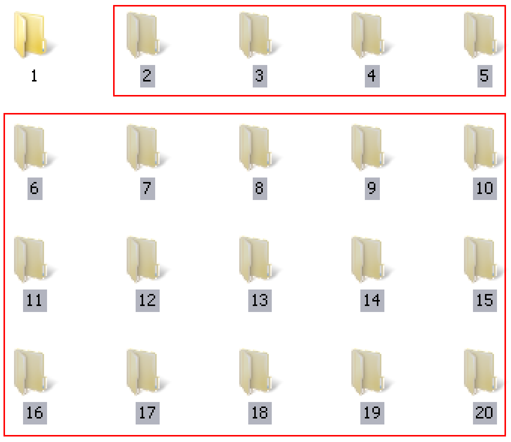
The images below illustrate statement tests.

The first test:



In this test we can select folders 3 and 4 with out first selection, folders 5, 6, 7, 8 with our second selection and folder 9 with our third, last selection.

The second test:



In this test we can first select all folders in the first row (2, 3, 4, 5), then — all other ones.

D. End of Exams

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

Students love to celebrate their holidays. Especially if the holiday is the day of the end of exams!

Despite the fact that Igor K., unlike his groupmates, failed to pass a programming test, he decided to invite them to go to a cafe so that each of them could drink a bottle of... fresh cow milk. Having entered the cafe, the m friends found n different kinds of milk on the menu, that's why they ordered n bottles — one bottle of each kind. We know that the volume of milk in each bottle equals w.

When the bottles were brought in, they decided to pour all the milk evenly among the m cups, so that each got a cup. As a punishment for not passing the test Igor was appointed the person to pour the milk. He protested that he was afraid to mix something up and suggested to distribute the drink so that the milk from each bottle was in no more than two different cups. His friends agreed but they suddenly faced the following problem — and what is actually the way to do it?

Help them and write the program that will help to distribute the milk among the cups and drink it as quickly as possible!

Note that due to Igor K.'s perfectly accurate eye and unswerving hands, he can pour any fractional amount of milk from any bottle to any cup.

Input

The only input data file contains three integers n, w and m ($1 \le n \le 50$, $100 \le w \le 1000$, $2 \le m \le 50$), where n stands for the number of ordered bottles, w stands for the volume of each of them and m stands for the number of friends in the company.

Output

Print on the first line "`YES`" if it is possible to pour the milk so that the milk from each bottle was in no more than two different cups. If there's no solution, print "`NO`".

If there is a solution, then print m more lines, where the i-th of them describes the content of the i-th student's cup. The line should consist of one or more pairs that would look like "b v". Each such pair means that v ($v > 0$) units of milk were poured into the i-th cup from bottle b ($1 \le b \le n$). All numbers b on each line should be different.

If there are several variants to solve the problem, print any of them. Print the real numbers with no less than 6 digits after the decimal point.

Sample test(s)

input
```
2 500 3
```
output
```
YES
1 333.333333
2 333.333333
2 166.666667 1 166.666667
```

input
```
4 100 5
```
output
```
YES
3 20.000000 4 60.000000
1 80.000000
4 40.000000 2 40.000000
3 80.000000
2 60.000000 1 20.000000
```

input
```
4 100 7
```
output
```
NO
```

input
```
5 500 2
```
output
```
YES
4 250.000000 5 500.000000 2 500.000000
3 500.000000 1 500.000000 4 250.000000
```

E. Azembler

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

After the Search Ultimate program that searched for strings in a text failed, Igor K. got to think: "Why on Earth does my program work so slowly?" As he double-checked his code, he said: "My code contains no errors, yet I know how we will improve Search Ultimate!" and took a large book from the shelves. The book read "Azembler. Principally New Approach".

Having carefully thumbed through the book, Igor K. realised that, as it turns out, you can multiply the numbers dozens of times faster. "Search Ultimate will be faster than it has ever been!" — the fellow shouted happily and set to work.

Let us now clarify what Igor's idea was. The thing is that the code that was generated by a compiler was far from perfect. Standard multiplying does work slower than with the trick the book mentioned.

The Azembler language operates with 26 registers (eax, ebx, ..., ezx) and two commands:

- $[x]$ — returns the value located in the address x. For example, [eax] returns the value that was located in the address, equal to the value in the register eax.
- lea x, y — assigns to the register x, indicated as the first operand, the second operand's address. Thus, for example, the "lea ebx, [eax]" command will write in the ebx register the content of the eax register: first the [eax] operation will be fulfilled, the result of it will be some value that lies in the address written in eax. But we do not need the value — the next operation will be lea, that will take the [eax] address, i.e., the value in the eax register, and will write it in ebx.

On the first thought the second operation seems meaningless, but as it turns out, it is acceptable to write the operation as

lea ecx, [eax + ebx],

lea ecx, [k*eax]

or even

lea ecx, [ebx + k*eax],

where k = 1, 2, 4 or 8.

As a result, the register ecx will be equal to the numbers eax + ebx, k*eax and ebx + k*eax correspondingly. However, such operation is fulfilled many times, dozens of times faster that the usual multiplying of numbers. And using several such operations, one can very quickly multiply some number by some other one. Of course, instead of eax, ebx and ecx you are allowed to use any registers.

For example, let the eax register contain some number that we should multiply by 41. It takes us 2 lines:

lea ebx, [eax + 4*eax] // now ebx = 5*eax

lea eax, [eax + 8*ebx] // now eax = eax + 8*ebx = 41*eax

Igor K. got interested in the following question: what is the minimum number of lea operations needed to multiply by the given number n and how to do it? Your task is to help him.

Consider that at the initial moment of time eax contains a number that Igor K. was about to multiply by n, and the registers from ebx to ezx contain number 0. At the final moment of time the result can be located in any register.

Input

The input data contain the only integer n ($1 \le n \le 255$), which Igor K. is about to multiply.

Output

On the first line print number p, which represents the minimum number of lea operations, needed to do that. Then print the program consisting of p commands, performing the operations. It is guaranteed that such program exists for any n from 1 to 255.

Use precisely the following format of commands (here k is equal to 1, 2, 4 or 8, and x, y and z are any, even coinciding registers):

lea x, [y]

lea x, [y + z]

lea x, [k*y]

lea x, [y + k*z]

Please note that **extra spaces at the end of a command are unacceptable**.

Sample test(s)

input
41
output

```
2
lea ebx, [eax + 4*eax]
lea ecx, [eax + 8*ebx]
```

input

```
2
```

output

```
1
lea ebx, [eax + eax]
```

input

```
4
```

output

```
1
lea ebx, [4*eax]
```