We can all benefit by doing occasional "toy" programs, when artificial restrictions are set up, so that we are forced to push our abilities to the limit. ... The art of tackling miniproblems with all our energy will sharpen our talents for the real problems.
Donald E. Knuth

Quarterfinal

Central region of Russia

Rybinsk, October 18-20-2016

Input file name	**INPUT.TXT**
Output file name:	**OUTPUT.TXT**
Memory: 64 Mb,	**Time: 1 sec / test**

A. Fried Fish

It is a well-known fact that Rybinsk once used to be a great fishing spot. People who lived in this town had fish for breakfast, lunch, and dinner. And whatever they could not eat themselves, they shipped to other towns and cities. Therefore, optimizing the fish cooking process was an important task.

Let us assume that today we caught ***N*** fish of the same size. Our frying pan fit up to ***K*** fish. Another assumption is that each side of each fish has to be fried for one minute. As a reminder, fish is usually fried on two sides.

And now let us count the minimum time we need to fry all the fish we have.

Limitations

$1 \le N, K \le 500$

Input

The input file contains two integer numbers ***N*** and ***K*** – the number of fish caught and the number of fish that can fit into our frying pan.

Output

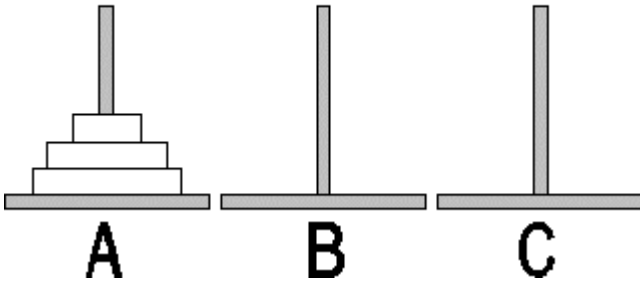
The output file must contain one integer number – the minimum time (minutes) needed to fry all the fish.

Examples

Input.txt	Output.txt
3 2	3
4 2	4

B. Hanoi tower

It has become a good tradition to solve the “Hanoi tower” puzzle at programming contests in Rybinsk. We will review the rules briefly.



There are 3 rods marked **A**, **B**, **C**. Initially, *N* disks of different diameter are placed on rod **A**: the smallest disk at the top, the disks below it are ordered by diameter, in increasing order. Rods **B** and **C** are empty yet.

The goal is to move all disks from rod **A** to rod **B**, using rod **C** as auxiliary.

At each step you can take the uppermost disk from any rod and then put it either on an empty rod or on a rod where the diameter of the uppermost disk is greater than the diameter of the disk taken.

Many books on programming give a recursive solution of this task. Here is a sample procedure in Pascal.

```
Procedure Hanoi (X, Y, Z: char; N: integer);
Begin
 If N>0 then
    Begin
       Hanoi (X, Z, Y, N-1);
       Writeln('Disk ', N, ' from ', X, ' to ', Y);
       Hanoi (Z, Y, X, N-1)
    End
End;
```

Your task is to write a program that will determine the number of the move, after which the disks for the first time after the start of the game are distributed equally between the rods.

Input

The input file contains an integer *N* – the number of disks.

Output

The output file must contain an integer number – the number of the move, after which the disks for the first time after the start of the game are distributed equally between the rods.

Limitations

1 <= ***N*** <=300 , ***N*** mod 3 = 0

Examples

Input	Output
3	2
6	9

C. Desktop

Vasily is a college student who likes to keep everything in order. One of his hobbies is software. Yesterday he installed a new operating system ***Windows 999999***. The key difference between the new system and ***Windows XP*** is that ***Windows 999999*** has 999,999 pre-installed utilities and after the installation of the system, the desktop is absolutely empty.

Vasily wants to place the maximum number of shortcuts to the utilities on his desktop. His computer screen is a grid sized $h \times w$ and the icons are squares sized 2×2. Icons must be positioned so that their corners match the borders of grid cells. They can also overlap each other, so the bottom icon will be partly covered by the top one. For comfort, Vasily decided that at least half of an icon (2 cells) must be visible; otherwise it would be hard to click on it. Now Vasily wants to know the maximum number of icons that can be placed on his desktop and the appropriate positioning pattern.

Limitations

$1 \le h, w \le 500$

Input

The first line in the input file contains two integer numbers ***h*** and ***w*** , representing the height and the width of Vasily's computer screen.

Output

The first line in the output file must contain the number ***N*** – the maximum number of icons that can be fitted on Vasily's desktop considering the conditions listed above. If $N>0$, the next ***N*** lines must contain pairs of numbers representing the coordinates of the upper right corners of the icons (line number and column number). The icons will be placed in the same order as they are listed, so each subsequent icon might lap over some of the previously listed icons.

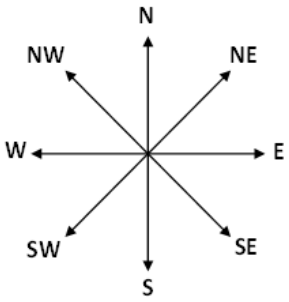
Examples

Input.txt	Output.txt
2 4	3 1 1 1 2 1 3

D. Weather Station

Albert is a well-known inventor. He's the one who designed an electronic weather station that periodically track all kinds of weather parameters and records the results of its measurements. While scanning the records made by the weather station, Albert discovered one important omission: wind direction data were recorded in one line without any separator characters. Albert became curious how many different solutions there would be if he tried to restore the original sequence of measurements.

The inventor wanted you to know that the station distinguishes eight different wind directions and encodes each one of them with one or two characters. In addition, he has drawn a picture with wind direction notations used by the weather station.



Your task is to write a program that will calculate the number of original sequences for a specific record based on weather station data. Albert realizes that the resulting number may be quite large, so your task is merely to calculate the value modulo $10^9 + 7$.

Limitations

The length of the input line does not exceed 10^5.

Input

The input file consists of a single line that contains a record made by the weather station. The record is a line of wind direction values containing characters N, S, W, and E.

Output

Output file must contain integer number of possible solutions modulo $10^9 + 7$.

Examples

Input.txt	Output.txt
NEWS	2
EWNS	1

Note

The line in the first example has two solutions: *{N, E, W, S}* and *{NE, W, S}*.

The line in the second example has one solution: *{E, W, N, S}*.

E. Cupcakes

In a college dormitory, somebody posted an announcement saying there would be a table with ***K*** cupcakes (***K***>=*0*) on the first floor of the building. Students thought it was a great idea and they lined up in a long queue to get a cupcake.

For each student number ***i***, there is a pre-defined maximum number of cupcakes (a_i) he or she could eat in one go. However, it does not mean every student will eat the maximum number of cupcakes. The actual quantity of cupcakes eaten could be any number in the range from 1 to the a_i. Once a student has taken his or her turn and eaten the cupcakes, he or she goes back to the end of the queue. This continues until all the cupcakes are gone. The first student who finds no cupcakes on the table when it is his or her turn will have to clean up after the party (if K=0, the first student in the queue has to clean the table).

Among the students, there is a Greedy Guy, who always eats exactly a_x cupcakes (if the number of cupcakes on the table is smaller than a_x, then he eats up all that is left). It is very easy to find out who the Greedy Guy is: his a_x is greater than all other a_i.

All the students (except the Greedy Guy) want the Greedy Guy to clean up. Your task is to help them determine if it is possible.

Limitations

$2 \le N \le 10^5$, $0 \le K \le 10^8$, $1 \le a_i \le 10^9$

There is always only one single student with the maximum value of a_i.

Input

The first line contains two inegers – ***N*** and ***K*** , representing the number of students and the number of cupcakes respectively. The next line contains ***N*** integer numbers a_i, representing the maximum number of cupcakes that can be eaten by student number *i* in the queue.

Output

The output is "YES" if the students can make relevant arrangements to achieve their goal and have the Greedy Guy clean the table. Otherwise the output is "KEK."

Examples

Input.txt	Output.txt
4 3 1 2 3 2	YES
5 8 1 2 3 2 1	KEK

F. Vitamins

A schoolboy named Vasya had been spending so much time on his computer that he became colorblind. His doctor prescribed him vitamins. When Vasya bought himself a bottle of vitamins, he found out they had to be taken three times a day – a white pill in the morning, a red one after lunch, and a blue one before going to bed. The problem was that all the pills were in the same bottle, and the boy could not tell the color of the pills. After reading the directions carefully, Vasya discovered that pills of different colors had different weights – the white ones were the heaviest, the red ones were lighter, and the blue pills were the most lightweight. Vasya arranged all the pills in a row and numbered them from 1 to n; then he took a pharmacy scale and started weighing the pills. For each measurement, the boy selected two pills, put them in the opposite weighing pans, and recorded the weighing results in a table. After ***m*** measurements, Vasya decided he had enough information to tell the color of each pill.Your task is to write a program that will determine the color of each pill based on a partially filled table of measurements.

Limitations

$3 \le n \le 1000$, $0 \le m \le n(n-1)/2$, $a_i \ne b_i$.

Input

The first line of the input file contains integer number ***n*** – the number of pills. The second line contains number ***m*** – the number of measurements taken. The following ***m*** lines each contain two integer numbers a_i and b_i – the numbers of weighed pills separated by characters '<,' '>' or '=,' representing the measurement results. For example, "2<3" means that the second pill is lighter than the third one; "2>5" means the second pill is heavier than the fifth one; "3=4" means the third and the fourth have the same weight.

Output

The output file must contain a single line consisting of ***n*** characters. The first character represents the color of the first pill, the second character represents the color of the second pill, and so on. White pills are designated as "**W**", red ones as "**R**" and blue ones as "**B**". In addition, pills are marked with the symbol "**?**" in case their color is uncertain.

Examples

Input.txt	Output.txt
3 2 1<3 1>2	RBW

G. Sphenic numbers

Schoolboy Vasya is interested in the problem of distinguishing prime numbers. He has decided to develop his own testing method for it.

Unfortunately, the new algorithm has one deficiency – it yields false positive outputs in case with so-called sphenic numbers. For those who does not know: sphenic number is the product of exactly three distinct prime numbers.

Help Vasya write a program to distinguish sphenic numbers.

Inut

The input file contains a single number – integer ***n***.

Limitations

$30 \le n \le 10467397$.

Output

The output file must contain one single line with the value "**YES**" (without quotation marks) if number ***n*** is sphenic or "**NO**" if it is not.

Examples

Input.txt	Output.txt
30	YES
40	NO
10467397	YES

H. Non-random numbers

Vasya is a schoolboy who was playing around with a random number generator and noticed that it never generates numbers with the value of a specific digit equal to the position of that digit in the number.

Vasya became curious and he came to discover the following:

- The input accepted by the generator is one positive integer ***n*** – the number of digits in the generated random number.
- The output is a positive integer number consisting of ***n*** digits without leading zeroes.
- In the generated number at ***i*** position (from the left-hand side) cannot be digit ***i***.

For example, if we want the generator to produce a single-digit number, it will generate any single-digit number except 0 or 1. In case with a double-digit number, the output will be anything except 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 22, 32, 42, 52, 62, 72, 82 or 92.

The schoolboy decided to find out how many different numbers can be generated for any given ***n***-digit number.

Your task is to write a program that will help the schoolboy to solve this problem.

Limitations

$1 \le n \le 100$.

Input

The input file contains single integer ***n*** – the number of digits in the generated random number.

Output

The output file must contain one single integer – the number of possible random ***n***-digit numbers. The output must have no leading zeroes.

Examples

Input.txt	Output.txt
1	8
2	72
12	344373768000

I. Land Division

Neleppo is a town lying on the border of two enemy states. On the map, it looks like an ***N***-sided convex polygon. No three vertices of the polygon are on the same line. To settle the numerous disputes and avoid further victims, the two states have signed an agreement which divides the town in two parts. One part must be a ***K***-sided polygon and the other, an ***M***-sided polygon. These parts will belong to different states. A fence must be built between the parts, stretching from one point on the town border to another such point. To minimize the costs, the fence must be linear.

Your task is to write a program that will calculate the minimum length of the fence satisfying the following conditions.

Limitations

$2 < N, M, K < 100$

$-1000 \le X_i, Y_i \le 1000$

Input

The first line of the input file contains numbers ***N***, ***M***, and ***K*** separated by a space. The next ***N*** lines each contain two integers X_i and Y_i representing coordinates of the polygon's vertices. The vertices are listed in the clockwise order.

Output

Length of the border fence accurate to three decimal places, or −1 if there is no solution.

Example

Input.txt	Output.txt
4 4 4 0 0 1 1 2 1 1 0	0.707

Explanations for the example shown in the figure below.

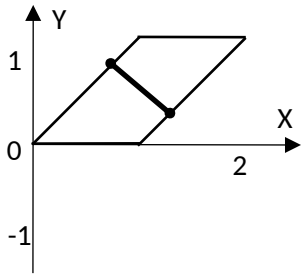


Fig. 1

J. Architect of Your Own Fortune

Vasya is a schoolboy who rides a trolley bus and then a bus to get to school. He's always very happy to get a "lucky ticket," which means the total of the first three digits in the ticket number equals the total of the last three digits. However, Vasya has been down on his luck ever since the beginning of the new school year – over the past month, he hardly had any lucky tickets at all.

Vasya thought this over and decided to take control of the situation. Upon reviewing his recent tickets, he realized he can produce several lucky numbers by combining trolley bus tickets with bus tickets. To that end, the schoolboy had to take two tickets, fold them in half along the vertical axis and join halves of different tickets together. The first three digits of the new "super lucky ticket" are the three digits on the left-hand side of the ticket for one mode of transport, and the last three digits are the three digits on the right-hand side of the ticket for the other mode of transport.

Example: let us assume that Vasya has a trolley bus ticket with the number **123456** and a bus ticket with the number **789222**. They can be combined either as **123222** or as **789456**. The first one of these two combinations is super lucky.

Your task is to write a program that will help produce the maximum number of super lucky combinations from the tickets available, assuming each ticket can only be used in one combination.

Limitations

$1 \le n, m \le 100$.

Input

The first line of the input file contains two integer numbers n and m separated by a space. The second line of the input file contains n six-digit integer numbers separated by a space, representing numbers of bus tickets. The third line of the input file contains m six-digit integer numbers separated by a space, representing numbers of trolley bus tickets.

Output

The first line of the output file must contain integer number ***k*** – the maximum possible number of super lucky combinations. The following ***k*** lines must contain these super lucky combinations. Each line with a super lucky combination must start with the Latin letters "AT" if the combination begins with numbers from a bus ticket and ends with numbers from a trolley bus ticket; otherwise the line must start with the letters "TA." The letters must be followed by two six-digit numbers separated by a space, representing the numbers of the relevant tickets.

Examples

Input.txt	Output.txt
2 2 123456 111222 141204 555000	2 TA 555000 123456 TA 141204 111222

K. Polymorphic code

Many computer viruses and worms use polymorphic code to hide their presence. Polymorphic code is a code each copy of which has a different set of operators, but performs the same function. Usually, polymorphic code is used to decrypt the main body of the virus or worm.

A kind of polymorphism is virus source code "psevdocompilation", when random virus body commands are replaced by an equivalent sequence of commands.

There is a code snippet compiled by "psevdocompiler" working according to the following algorithm:

1. If the code length exceeds some constant, then the algorithm stops

2. If the code length does not exceed, then a random command chosen and replaced by a sequence of commands.

The following commands can be used in a polymorphic code:

Command	Psevdocode	Command description
MOV Rx, Ry	Rx ← Ry	Move data contained in the Ry register into the Rx register
MOV Rx, const	Rx ← const	Load constant into the Rx register
ADD Rx, Ry	Rx ← (Rx + Ry) mod 256	Addition of register Rx and Ry modulo 256. The result is placed in the Rx register
ADD Rx, const	Rx ← (Rx + const) mod 256	Adding to the register Rx constant *const* modulo 256
XOR Rx, Ry	Rx ← Rx ⊕ Ry	Exclusive disjunction of data contained in Rx and Ry registers. The result is placed in the Rx register.
XOR Rx, const	Rx ← Rx ⊕ const	Exclusive disjunction of data contained in Rx register and constant
PUSH Rx	STop ← STop + 1 Stack[STop] ← Rx	Adds data contained in the Rx register to the stack
POP Rx	Rx ← Stack[STop]	Remove the top of the stack into the Rx register

	STop ← STop – 1	

Rx or Ry denotes one of the registers R1, R2, ..., R8, STop – top of the stack, and a Stack [] – the stack itself. Constants “const “are hexadecimal constants in the range 0 ... FF.

To convert, "psevdocopmiler" uses the following replacements:

Replacable Command	Replacement	Comment
MOV Rx, Ry	PUSH Ry POP Rx	
MOV Rx, const	XOR Rx, Rx ADD Rx, const	

ADD Rx, Ry	PUSH Rz
	MOV Rz, Ry
	ADD Rx, Rz
	POP Rz
		Rz ≠ Rx

ADD Rx, const	PUSH Rz
	MOV Rz, const
	ADD Rz, Rx
	MOV Rx, Rz
	POP Rz
		Rz ≠ Rx

XOR Rx, Ry	PUSH Rz
	MOV Rz, Ry
	XOR Rz, rndConst
	XOR Rx, Rz
	MOV Rz, rndConst
	XOR Rx, Rz
	POP Rz
		Rz ≠ Rx
		rndConst = Random(01... FF)

XOR Rx, const	PUSH Rz	Rz ≠ Rx

	XOR Rz, const XOR Rx, Rz POP Rz XOR Rx, Rz	
PUSH Rx	PUSH Rx MOV Rx, Rz POP Rz PUSH Rz XOR Rx, Rz XOR Rz, Rx XOR Rx, Rz	Rz ≠ Rx
POP Rx	MOV Rx, Rz POP Rz XOR Rx, Rz XOR Rz, Rx XOR Rx, Rz	Rz ≠ Rx

Your task will be to generate the source code based on the code formed by "psevdocompiler". If there are several such codes, find the shortest one.

Limitations

The length of the program formed by "psevdocompiler" does not exceed 50,000 commands:

$0 < \boldsymbol{n} \le 50{,}000$.

Input

The first line contains integer ***n*** – the number of commands. Then follow n commands. Each command is placed on a separate line, in capital letters, without leading spaces. Parameters are separated from the command by one space. If the command contains two parameters, then a comma and a space is placed after the first parameter.

Output

The first line of the output file contains integer ***k*** – the number of commands in the source code. Then ***k***-commands of restored source code are followed in the same format, which they are specified in the input file.

Examples

Input.txt	Output.txt
5 PUSH R2 PUSH R6 POP R2 ADD R3, R2 POP R2	1 ADD R3, R6

5
PUSH R5
MOV R5, 3E
ADD R5, R3
MOV R3, R5
POP R4
	5
	PUSH R5
	MOV R5, 3E
	ADD R5, R3
	MOV R3, R5
	POP R4

Rybinsk State Aviation Technical University

Program Committee

- Sergey G. Volchenkov,
- Vladimir N. Pinaev,
- Michael Y. Kopachev,
- Andrey Mirzoyan,
- Sergei Dobrikov,
- Dmitry Shalaev