Problem 1 – Next Date

We are given a date (day + month + year). Write a program to print the next day.

Input

The input data consists of 3 lines holding the integer numbers: **day**, **month** and **year**.

The input data will always be valid and in the format described. There is no need to check it explicitly.

Output

The output data should be printed on the console in the format **day.month.year** (no leading zeroes).

Constraints

- The number **day** is in the range [1…31] inclusive.
- The number **month** is in the range [1…12] inclusive.
- The number **year** is in the range [2000…2013] inclusive.
- The date is valid according to the classical calendar system.
- Allowed work time for your program: 0.1 seconds.
- Allowed memory: 16 MB.

Examples

Input	Output
1 11 2012	2.11.2012

Input	Output
30 9 2011	1.10.2011

Input	Output
28 2 2003	1.3.2003

Input	Output
31 12 2012	1.1.2013

Problem 2 – Tribonacci Triangle

You all know the Fibonacci sequence. Well, the Tribonacci sequence is almost the same, but it uses the last three numbers (instead of the last two) to calculate the next number in the sequence. So, we can define each element in the sequence as:

$$T_n = T_{n-1} + T_{n-2} + T_{n-3}$$

where T_n is the current Tribonacci number (n is the index of the current Tribonacci number).

The Tribonacci sequence can begin with any three integer numbers – positive or negative – and continue as described by the formula above.

Now, a Tribonacci triangle is a triangle of numbers from the Tribonacci sequence. The first line of the triangle contains only the first number of the Tribonacci sequence. The second line contains the second and third numbers of the Tribonacci sequence, separated by a single whitespace (" "). The third line contains the next three numbers of the Tribonacci sequence (again, separated by whitespaces). The fourth line contains the next four numbers and so on. Basically, every line contains one more number than the previous.

Your task is to write a program, which prints to the console a Tribonacci triangle by given the first three numbers of the Tribonacci sequence, and the number of lines in the triangle.

Input

The input data should be read from the console.

The **first three lines** will contain the values of the **first three numbers** of the Tribonacci sequence – each number will be on a separate line.

On the fourth line of the input there will be the number **L** – **the number of lines** in the Tribonacci triangle.

The input data will always be valid and in the format described. There is no need to check it explicitly.

Output

The output data should be printed on the console.

The output should contain **exactly L lines**. The **first line should contain exactly one number**, the **second line – exactly two numbers**, the **third line** (if L>2) **– exactly three numbers**, …, the **L-th line should contain exactly L numbers**. Numbers should be **separated by exactly one whitespace** (" "), and there **shouldn't be any whitespaces after the last number on a line**.

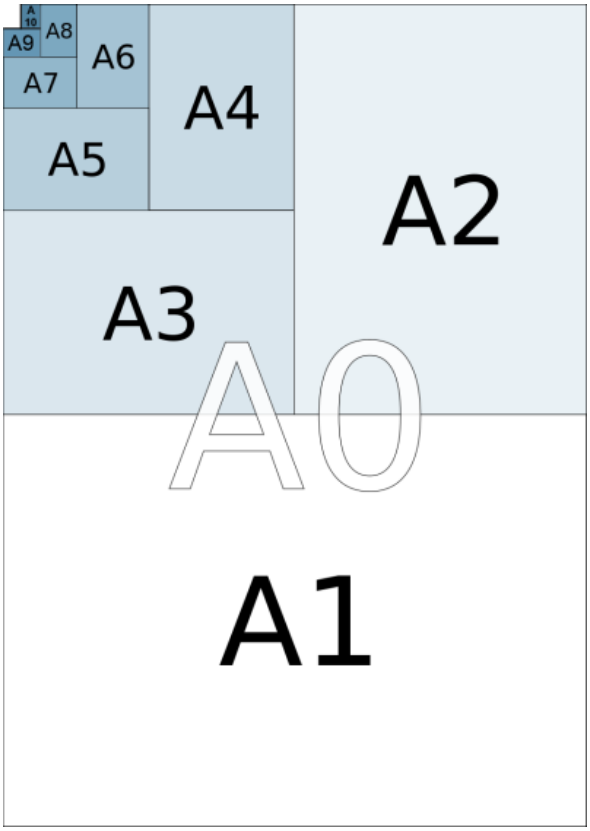
Constraints

- $2 \leq L \leq 20$.
- Any number in the Tribonacci triangle can be stored in a 64-bit signed integer.
- Allowed working time for your program: 0.1 seconds. Allowed memory: 16 MB.

Examples

Input example	Output example
1 2 3 3	1 2 3 6 11 20
1 -1 1 4	1 -1 1 1 1 3 5 9 17 31

Problem 3 – Sheets



Asya loves confetti. One day she decided to create exactly **N** small **pieces** of sheets **with paper size A10**.

A10 is a standard for paper sizes. A9 is another standard that is **twice as bigger** as A10, so A9 can be **cut** into 2 pieces of size A10. A8 is twice as bigger as A9 and so on. A0 is twice as bigger as A1. See the picture on the left.

Asya has **only one sheet of each type** (totally 11 sheets). She wants to have **exactly N pieces** of size A10 by cutting **as few sheets as possible**.

Asya should not have any wasted sheets.

Write a program for her.

For example if we want to cut sheets into 9 pieces with the size of A10, we will use the only A7 sheet (cut into 8 pieces of size A10) and the only sheet with size A10. Then we will use 2 sheets. All other 9 sheets will not be used.

Input

On the only line of the input there will be the number N.

The input data will always be valid and in the format described. There is no need to check it explicitly.

Output

Print the sizes of the sheets that **will not be used** after Asya's cutting. Print one size on a single line.

The order of the paper sizes **doesn't** matter. See the examples below.

Constraints

- **N** will be between 0 and 2046, inclusive.
- Allowed work time for your program: 0.1 seconds. Allowed memory: 16 MB.

Examples

Example input	Example output
1	A9 A8 A7 A6 A5 A4 A3 A2 A1 A0

Example input	Example output
9	A9 A8 A6 A5 A4 A3 A2 A1 A0

Example input	Example output
666	A0 A10 A2 A5 A8 A4

Example input	Example output
1337	A1 A3 A4 A8 A9

Problem 4 – Carpets

Telerik Academy is considering opening a new office in Great Britain. Therefore the whole Trainers team is traveling to the United Kingdom for the important event. Of course all of them want to feel exactly like home in the new office, so they ordered some special carpets from Chiprovtsi. Those carpets consist of many embedded rhombs. Please help them and print some carpets in different sizes for the new Telerik Academy Head Quarters.

Input

The input data should be read from the console.

You have an integer number N (always **even** number) showing the **width** and the **height** of the most outer rhomb. The **width** and the **height** will always be **equal**.

The input data will always be valid and in the format described. There is no need to check it explicitly.

Output

The output should be printed on the console.

Use the "/" and "\" characters to print the rhomb sides and "." (dot) for the rest. You should print exactly one space between each rhomb.

Constraints

- **N** will always be a positive **even** number between **6** and **80** inclusive.
- Allowed working time for your program: 0.1 seconds.
- Allowed memory: 16 MB.

Examples

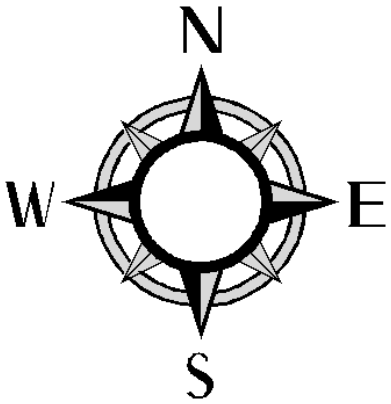
Example input	Example output
6	.. ./ \. / /\ \ \ \/ / .\ /. ..\/..

Example input	Example output
12	..../ \.... .../ /\ \... ../ / \ \.. ./ / /\ \ \. / / / \ \ \ \ \ \ / / / .\ \ \/ / /. ..\ \ / /.. ...\ \/ /... ....\ /.... .....\/.....

Problem 5 – Formula Bit 1

The residents of Bitlandia are huge sports fans. The bits have played almost every single sport that they have learned from watching human TV i.e. EuroBitSport and BitTV. Today for the first time they watched Formula 1 and now they certainly want to build a local track and start practicing right away. Of course the bits don't want to copy the humans. They want to be unique and therefore they've added some special rules:

1. The **track** must be built on a **grid** of **8x8 cells**, containing only zeros and ones.
2. The track itself must contain only **zeros**. The width of the track will be **one cell**.
3. The track must start from the **upper right corner** and end on the **lower left corner**.
4. The cars can move only in 3 directions – **South** (down), **West** (left) And **North** (up).
5. The first direction must always be **south**.
6. The cars must move in the current direction, while it is possible i.e. the cars can turn only when it reaches the **end** of the **grid** or a **cell**, containing a bit with a value of **one**.
7. The cars can switch between directions only in the following order:
   South -> West -> North -> West (and then again South -> West and so on)



You will receive information about the grid as a list of **8** bytes (positive integers in the range [0…255]) n_0, n_1, …, n_7. The grid itself is represented by the bits of those bytes.

Your task is to find whether a track can be built on the given grid. If the grid is appropriate, you should print the length of the track and the count of the turns in it (the switches between directions), otherwise you should print "No" and the length of the track until it was interrupted.

Input

The input data should be read from the console.

There will be exactly 8 lines each holding an integer number (n_0, n_1… n_7).

The input data will always be valid and in the format described. There is no need to check it explicitly.

Output

The output data should be printed on the console.

On the only output row you should print two numbers in the following format "X Y", where X is the length of the track and Y is the count of the turns. If a track cannot be built, you should print "No X", where X is the length of the track, until it was interrupted.

Constraints

- The numbers $\mathbf{n_0}$, $\mathbf{n_1}$, …, $\mathbf{n_7}$ are positive integers in the range [0…255]
- Allowed work time for your program: 0.1 seconds.
- Allowed memory: 16 MB.

	7	6	5	4	3	2	1	0	
0	**0**	**0**	**0**	**0**	**0**	0	1	**0**	$n_0 = 2$
1	**0**	0	1	0	**0**	1	1	**0**	$n_1 = 38$
2	**0**	0	0	1	**0**	1	0	**0**	$n_2 = 20$
3	**0**	0	1	1	**0**	**0**	**0**	**0**	$n_3 = 48$
4	**0**	1	1	0	1	1	1	1	$n_4 = 111$
5	**0**	0	0	0	1	1	1	1	$n_5 = 15$
6	**0**	0	0	0	0	0	1	1	$n_6 = 3$
7	**0**	0	1	0	1	0	1	1	$n_7 = 43$

	7	6	5	4	3	2	1	0	
0	0	0	0	0	0	0	1	**0**	$n_0 = 2$
1	0	0	1	0	0	1	1	**0**	$n_1 = 38$
2	0	0	0	1	1	1	0	**0**	$n_2 = 28$
3	0	0	0	1	**0**	**0**	**0**	**0**	$n_3 = 16$
4	0	1	0	0	0	1	1	1	$n_4 = 71$
5	1	0	0	0	1	1	1	1	$n_5 = 143$
6	0	0	0	0	0	0	1	1	$n_6 = 3$
7	0	0	1	0	1	0	1	1	$n_7 = 43$

Examples

Example input	Example output
2 38 20 48 111 15 3 43	21 4

Example input	Example output
2 38 28 16 71 143 3 43	No 7