Telerik Software Academy 2012/2013 – C# Part 1 – Practical Exam – Variant 3

Exam terms – Read carefully!

During the exam you **are allowed** to use any teaching materials, lectures, books, existing source code, and other paper or Internet resources.

Direct or indirect communication with anyone in the class or outside the room is absolutely **forbidden**.

The address of the judge system is: http://bgcoder.com. Trainers will give you the contest password. Only your **last** code submission will be evaluated. All decimal separators in all tasks will be "." (Dot).

You have exactly **6 hours** to solve all given problems. Good luck!

After the end of the exam you should leave all used papers (including this one) in the room!

Problem 1 – Triple Rotation of Digits

In Kaspichan we drink a lot. One evening we drunk enough so we invited an interesting game: Someone says a number. The first person after him rotates the digits of this number by moving its last digit at its start (e.g. 12345 → 51234). The second person after that again rotates the number (e.g. 51234 → 45123). Finally the third person after him also rotates the number (e.g. 45123 → 34512). The obtained number then is sent by SMS to a fellow group of alcoholics who continue the game at their drink place.

Write a program that helps the Kaspichan drinkers to calculate the triple digits rotation of given number **K**. Note that zeroes could also take part in the play and the leading digits are lost after each rotation, e.g. the triple rotation of 180001 is 1180 (180001 → 118000 → 011800 → 11800 → 01180 → 1180).

Input

The input data should be read from the console and consists of a single line holding an integer number **K**.

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 consist of a single line holding the number obtained after applying a triple digits rotation of the number **K**.

Constraints

- The number **K** is in the range [1…999 999] inclusive.
- Allowed work time for your program: 0.1 seconds. Allowed memory: 16 MB.

Examples

Input	Output
51234	23451

Input	Output
180001	1180

Input	Output
443	443

Input	Output
53	35

Problem 2 – Quadronacci Rectangle

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

$$Q_n = Q_{n-1} + Q_{n-2} + Q_{n-3} + Q_{n-4}$$

where Q_n is the current Quadronacci number (n is the index of the current Quadronacci number).

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

Now, a Quadronacci rectangle is what you probably expect – a rectangle (matrix) of numbers from the Quadronacci sequence. So we can say that the rectangle's height is actually the number of rows and the rectangle's width is the numbers of columns of numbers.

If use **R** for the number of rows and **C** for the number of columns, then the first row in the rectangle contains the first **C** numbers from the sequence, the second row contains the next C numbers from the sequence and so on.

Your task is to write a program, which prints to the console a Quadronacci rectangle by given the first four numbers of the Quadronacci sequence, the number of rows and the number of columns in the rectangle.

Input

The input data should be read from the console.

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

On the fifth line there will be the number **R** – the number of rows of the Quadronacci rectangle.

On the sixth line there will be the number **C** – the number of columns of the Quadronacci rectangle.

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 R lines** with **exactly C numbers per line** – representing each line of the Quadronacci rectangle. **Numbers should be separated by exactly one whitespace** (" "), and there **shouldn't be any whitespaces after the last number on a line**.

Constraints

- $1 \leq \mathbf{R} \leq 20$.
- $4 \leq \mathbf{C} \leq 20$.
- Any number in the Quadronacci rectangle 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 4 2 8	1 2 3 4 10 19 36 69 134 258 497 958 1847 3560 6862 13227
5 -5 1 2 3 4	5 -5 1 2 3 1 7 13 24 45 89 171

Problem 3 – Poker

Mitko is a famous poker player. But he is not a good developer. One day he found interesting **5-card poker game** which is played by a deck of 52 cards. Nothing strange here, except that the **cards have no suit (color)**. Mitko decided to start playing this game, but he needs a program to show him the best hand that he has. Write a program to help Mitko. The program should read the cards and check each of the following conditions and display one of the messages:

- If the five cards are equal, the program outputs “Impossible”, otherwise:
- if four of them are equal, the program outputs “Four of a Kind”, otherwise:
- if there are three equal and the another two are also equal (example: ‘2’, ‘2’, ‘10’, ‘10’, ‘2’), the program outputs “Full House”, otherwise:
- if the five are consecutive (example: ‘2’, ‘3’, ‘4’, ‘5’, ‘6’), the program outputs “Straight”, otherwise:
- if three of them are equal, the program outputs “Three of a Kind”, otherwise:
- if two pairs contain respectively equal numbers (example ‘A’, ‘K’, ‘A’, ‘K’, ‘J’), the program outputs “Two Pairs”, otherwise:
- if only two cards are equal, the program outputs “One Pair”, otherwise;
- the program outputs “Nothing”.

Cards are given with one of the following strings ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’, ‘9’, ‘10’, ‘J’, ‘Q’, ‘K’ or ‘A’ (without the quotes). The combination of ‘10’, ‘J’, ‘Q’, ‘K’ and ‘A’ are considered as a consecutive cards. Note that ‘A’, ‘2’, ‘3’, ‘4’ and ‘5’ are also consecutive cards so your program should output “Straight” in this case.

Input

The input data should be read from the console.

You will be given exactly 5 lines with one card per line.

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 line, write an appropriate message with the exact letters case.

Constraints

- Allowed working time for your program: 0.1 seconds. Allowed memory: 16 MB.

Examples

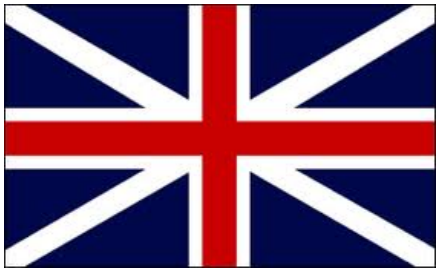
Example Input	Example output
2 7 A J K	Nothing

Example input	Example output
4 2 A 3 5	Straight

Example input	Example output
8 J J 8 8	Full House

Problem 4 – UK Flag

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. They've decided that everyone needs to bring the UK flag with him, as a token of respect to the local citizens. Please help them and print some flags in different sizes, so they will be well received. As a little reminder, here it is the flag itself:



Input

The input data should be read from the console.

You have an integer number N (always **odd** number) showing the **width** and the **height** of the flag. The flag will always be a **square**.

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 “*” (asterisk) character for the middle, the “\”, “/”, “|” (vertical dash), “-” (dash) characters for the lines and “.” (dot) for the rest.

Constraints

- **N** will always be a positive **odd** number between **5** and **79** inclusive.
- Allowed working time for your program: 0.1 seconds. Allowed memory: 16 MB.

Examples

Example input	Example output
5	\.|./ .|/. --*-- ./|\. /.|.\

Example input	Example output
9	\...|.../ .\..|../. ..\.|./.. ...|/... ----*---- .../|\... ../.|.\.. ./..|..\. /...|...\

Problem 5 – Angry Bits

The residents of Bitlandia love playing games. They try all sorts of games – outdoor games, team games, computer games, but their newest passion is the mobile gaming. All of the bits in Bitlandia compete in the game of Angry Bits. It's a game played on a grid of 16x8 cells, containing **zeros** and **ones**. All of the **ones** on the **left** side (columns 8-15) denote a **bird** and all of the **ones** on the **right** side (columns 0-7) denote a **pig**. The object of the game is to destroy the pigs. To achieve this, the player throws the birds towards them. The player wins if he destroys all of the pigs. He also gets a score according to the destroyed pigs. There are some special rules:

1. There will be no more than one bird in each column on the left side. There could be zero birds in some columns though.
2. There could be any number of pigs in each column on the right side (maximum 8 per column).
3. The birds are thrown in a special order, from right to left i.e. first the bird from column 8, then the one from column 9 and so on (considering that there are birds in those columns).
4. Each bird flies diagonally upwards and rightwards. When it reaches the top row, it starts flying diagonally downwards. Refer to the table. When the bird from cell [4, 8] is thrown it flies upwards to [0, 4], then starts flying downwards. (Note that if a bird is on the top row, it starts the flight diagonally downwards and rightwards)
5. A hit encounters when a bird reaches a cell with a pig in it. When that happens, the pig is destroyed along with all other pigs from the neighbor cells (horizontally, vertically or diagonally). Following the previous example, when the bird hits the pig in [2, 2], the pig is eliminated along with all six pigs near it.
6. The player wins points after each hit. The score for the hit is calculated according to the following formula: Length of the flight * Count of destroyed pigs. The length is measured in cells and does not include the starting cell. In our example the points for the hit are 42 (the flight is 6 cells long and the destroyed pigs are 7).
7. A bird ends its flight, when it hits a pig or when it reaches the right or bottom side of the field.
8. When a bird ends its flight, it is removed from the field along with all destroyed pigs i.e. their cells become zeros.

	15	*14*	*13*	*12*	*11*	*10*	*9*	*8*	*7*	*6*	*5*	*4*	*3*	*2*	*1*	*0*	
0	0	0	**0**	0	0	0	0	0	0	0	0	**0**	0	0	0	0	$n_0 = 0$
1	0	**0**	0	**0**	0	0	0	0	0	0	**0**	0	**0**	1	1	0	$n_1 = 6$
2	**1**	0	0	0	**0**	0	0	0	0	**0**	0	0	1	**1**	0	0	$n_2 = 32780$
3	0	0	0	0	0	**0**	0	0	**0**	0	0	0	1	1	1	0	$n_3 = 14$
4	0	0	0	0	0	0	**0**	**1**	0	0	0	0	0	0	0	0	$n_4 = 256$
5	0	0	0	0	0	0	0	**0**	0	0	0	0	0	0	0	0	$n_5 = 0$
6	0	0	0	0	0	0	0	0	**0**	0	0	0	0	0	0	0	$n_6 = 0$
7	0	0	0	0	0	0	0	0	0	**1**	0	0	0	0	0	0	$n_7 = 64$

You will receive information about the game field as a list of 8 ushorts (positive integers in the range [0…65535]) n0, n1, …, n7. The grid itself is represented by the bits of those ushorts.

Your task is to find the total score of the player and whether he wins or not. The player wins if all pigs are destroyed.

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 the final score followed by the words **“Yes”** or **“No”** depending on whether the player wins or not. They should be separated by a single whitespace.

Constraints

- The numbers $\mathbf{n_0}$, $\mathbf{n_1}$, …, $\mathbf{n_7}$ are positive integers in the range [0…65535]
- Allowed work time for your program: 0.1 seconds.
- Allowed memory: 16 MB.

Examples

Example input	**Example output**
0 6 32780 14 256 0 0 64	51 Yes

Example input	**Example output**
0 1 0 0 0 0 0 265	8 No