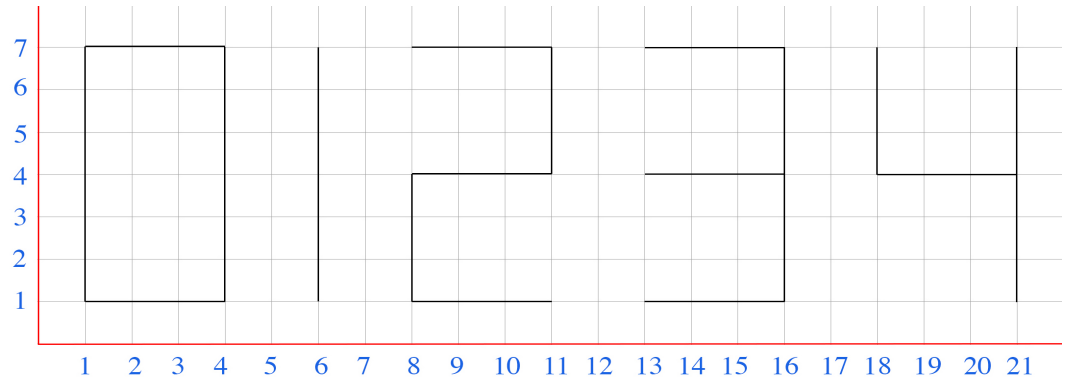
Problem E. Scrambled Digits

Program:	`digits.(cpp|java)`
Input:	`digits.in`
Balloon Color:	`Purple`

Little Petya started to learn how to write some digits, he is just learning to write the following 5 digits:



He draws each digit exactly with the same dimensions as the above image, but he might make zero or more of the following updates:

- Shrink or expand the width of the digit, keeping the width as an integer number of units.
- Shrink or expand the height of the digit, keeping the height as an integer number of units
- Rotate the digit clockwise by 90, 180 or 270 degrees.

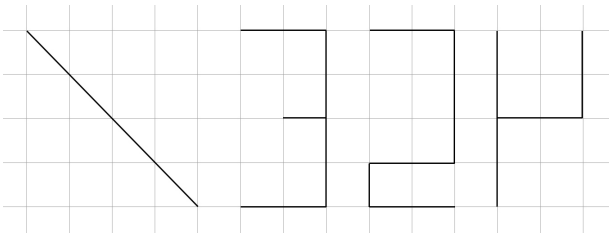
Note that shrinking and expanding a digit doesn't change the thickness of a segment, and it affects the whole digit equally. For example shrinking or expanding the digit 3 vertically (before rotating it), will always keep the middle horizontal segment exactly in the middle between the other horizontal segments.

Now he has an infinite 2D plane, where he writes a lot of digits, but he keeps the following rules valid:

- The digits won't touch each other, or overlap (the segments of each digit won't touch the segments of other digits).
- All vertical segments will be placed on a vertical line with an integer X-value.
- All horizontal segments will be placed on a horizontal line with an integer Y-value.
- All end points will be on a point with integer coordinates (an end point is a point at the end of a specific vertical or horizontal segment).

Petya will draw the digits by drawing line segments, each segment starts and ends at points with integer coordinates, and all segments are guaranteed to be vertical or horizontal with positive integer length. Note that the segments can touch or even overlap, but when this happens, these segments belong to the same instance of the same digit (check the second sample test case).

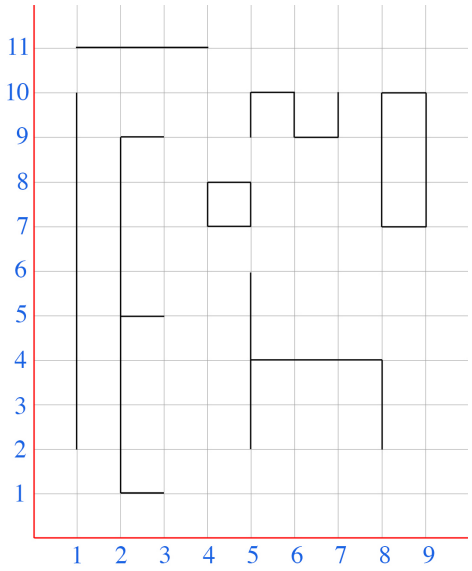
The following image represents some invalid digits which will never appear in the input:



The 1 is rotated incorrectly, the length of the middle horizontal segment in the 3 is wrong, the middle horizontal segment in the 2 isn't in the middle anymore and the 4 is mirrored (not rotated).

You will be given a list of line segments which Petya draw, and it's guaranteed that these line segments will form only valid digits as described above. Your task is to find how many times each digit was drawn.

- The first sample test case represents the first image.
- The second sample test case represents the following image (note that some segments might overlap and/or touch to form one longer segment):



Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer **T**, the number of test cases ($1 \leq \mathbf{T} \leq 100$). Followed by **T** test cases.

Each case starts with a line containing an integer **N** the number of segments to draw ($1 \leq \mathbf{N} \leq 100,000$). Followed by **N** lines, each line contains 4 integers, **X1**, **Y1**, **X2** and **Y2**, representing a line segment between the points (**X1**, **Y1**) and (**X2**, **Y2**) ($1 \leq \mathbf{X1}, \mathbf{Y1}, \mathbf{X2}, \mathbf{Y2} \leq 1,000,000,000$).

Output

For each test case, print 5 integers separated by a single space, each integer represents how many times each digit was drawn, in the same order from left to right like the first image.

Example

digits.in	standard output
2 17 1 1 1 7 1 7 4 7 4 7 4 1 4 1 1 1 6 1 6 7 11 1 8 1 8 1 8 4 8 4 11 4 11 4 11 7 11 7 8 7 13 1 16 1 16 1 16 7 16 7 13 7 16 4 13 4 21 1 21 7 21 4 18 4 18 4 18 7 24 1 2 1 4 1 4 1 8 1 6 1 10 1 11 4 11 2 1 2 9 2 1 3 1 2 5 3 5 2 9 3 9 4 7 4 8 4 8 5 8 5 8 5 7 5 7 4 7 5 2 5 6 5 4 8 4 8 4 8 2 5 9 5 10 5 10 6 10 6 10 6 9 6 9 7 9 7 9 7 10 8 7 8 10 8 10 9 10 9 10 9 7 9 7 8 7	1 1 1 1 1 2 2 1 1 1