Problem K. Surround the Cat

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	512 mebibytes

This is an interactive problem.

Consider a hexagonal tiling on the plane, as shown in the picture below.

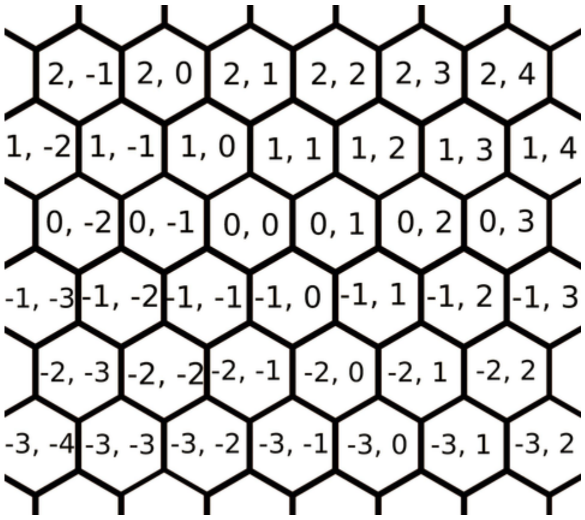
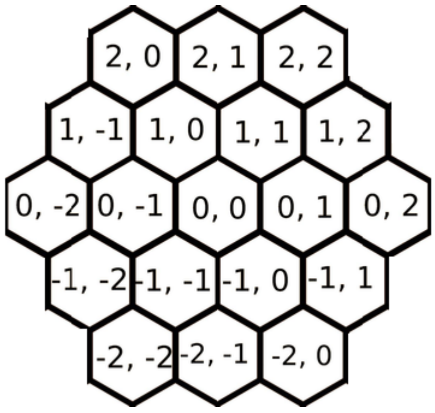


Fig. 1: What the plane looks like.

The house you built for your cat is a regular hexagon of side $N = 10$. Its six corners are located at $(9, 0)$, $(9, 9)$, $(0, 9)$, $(-9, 0)$, $(-9, -9)$, and $(0, -9)$.

For example, for $N = 3$ the house would look like this:



Now the cat wants to escape. Thus, you need to place some rocks in the house to surround the cat so that it can not escape.

Initially, the cat is located at $(0, 0)$. Every second, you can choose a location in the house and place a rock. Note that you cannot place a rock at the cat's current location. After that, the cat will choose a location that is adjacent to its current location and does not contain a rock, and move there.

Once the cat reaches the boundary of the house, it will escape and win. On the other hand, if the cat cannot choose a valid move, you win. Try to surround the cat so that it will not escape.

Interaction Protocol

Use standard input to read the locations of the cat. Use standard output to write the locations where you place the rocks. Each location is given on a single line containing two space-separated integers: the coordinates of the location.

Initially, the cat is at $(0, 0)$, and gives you "`0 0`" as input. You respond by printing the location of the first rock, then read the next location where the cat moved, and so on.

If you print an invalid location (it is not in the house, or currently contains the cat), terminate your program to get the "`Wrong Answer`" verdict. Placing a rock in a location that already contains a rock is allowed, but does not have any additional effects. If the cat arrived at the boundary of the house, you lose: terminate your program to get the "`Wrong Answer`" verdict. If the cat has no valid moves, you win: instead of reading the cat's next location, just terminate your program gracefully.

Don't forget to print the newline character and flush the output after printing each location: otherwise, you will likely get the "`Idleness Limit Exceeded`" verdict.

Example

standard input	*standard output*
0 0	
	1 1
1 0	
	0 1
0 0	
	-1 0
1 0	
	-1 -1
0 0	
	0 -1
1 0	
	9 9
0 0	
	1 0