Renting a PG

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	256 megabytes

Most of your seniors will go out for their summer internships this summer. With that comes the hassle of renting a PG. Your seniors have figured out the order in which they will choose a PG after considering the reviews they got. Students are smart but the PG owners are smarter. They have already figured out which students are more likely to revolt against their bland food so they have their own rank lists. Even though it's cruel, being the owners they have the right to turn down a student if a better one approaches. Assume that only one student can stay in one PG.
Your seniors are busy with the semester and they are hoping that you might be able to help them out with a solution so that it saves everyone's time.
Your goal is to figure out 'n' sets of Student-PG groups such that if a student 's' prefers a PG 'p' more than their current PG, then the owners of the PG 'p' should prefer their current student more than 's'.

Input

The first line contains an integer 't' which indicates the number of test cases.
For each test case, the first line contains an integer 'n'<=500 indicating the number of students and PGs.
The next 'n' lines are the preferences of the PG owners: i^{th} line contains the number i (which means that this is the preference list of the i^{th} owner) and the number of students in preferred order (first choice of i^{th} owner, second choice …). After this, in the next 'n' lines the preferences of the students follow in a similar format.
Note that both PG numbers and student numbers start from 1.

Output

For each test case print 'n' lines in which each line contains two number 's' and 'p' which means that student 's' should stay in PG 'p'. Sort the output with respect to student numbers.
The format should be like:
1 (Student 1's PG)
2 (Student 2's PG)
…
n (Student n's PG).
If no such combination exists then print -1.

Example

standard input	standard output
1	1 3
4	2 2
1 4 3 1 2	3 1
2 2 1 3 4	4 4
3 1 3 4 2	
4 4 3 1 2	
1 3 2 4 1	
2 2 3 1 4	
3 3 1 2 4	
4 3 2 4 1	