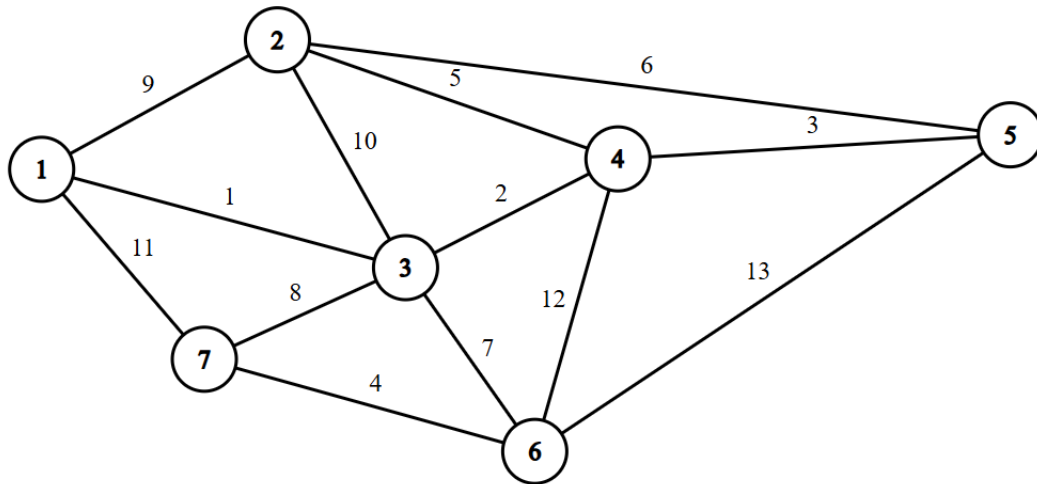


Forest trails

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

You are visiting a local forester and notice a map of the nearby forest.



The map shows intersections numbered from 1 to 7 and forest trails numbered from 1 to 13.

The local foresters have come up with their own algorithm for exploring the forest and written it as a function called **FOREST**:

```
FOREST(C, D)
  dist[C] = D
  for all trails of the form (C, X) or (X, C)
    if dist[X] not initialized
      FOREST(X, dist[C] + 1)
    else
      dist[C] = min(dist[C], dist[X] + 1)
```

In this function:

- **dist** — an array of distances whose values are initially undefined;
- **min** — the minimum function;
- **for all trails** — a loop through all trails connected to the current intersection;
- trails are **always processed in increasing order** of their numbers.

A detailed example of how this function works on the given map can be found in the **Note** section.

The foresters often use this algorithm to explore the forest without paying attention to the calculated distances.

However, some foresters also use it to find the shortest distance from the main entrance (intersection 1) to every other intersection.

Here, the distance is the minimum number of trails needed to reach an intersection, since every trail has length 1.

To calculate these distances, they call the function **FOREST(1, 0)**.

After looking at this function, you and the local forester realize that it does not always calculate the shortest distances correctly.

To demonstrate this, you decide to construct t example maps.

The i -th map must contain n_i intersections, and the function `FOREST(1, 0)` must calculate the correct shortest distance for **exactly** k_i of them.

For some maps, the number of trails is also fixed and must be equal to m_i .

More details about the requirements for these maps can be found in the **Output** section.

Input

The first line contains one positive integer t ($1 \leq t \leq 500$) — the number of test cases.

Each of the next t lines contains three integers n_i , m_i , and k_i — the requirements for the i -th map ($1 \leq n_i \leq 300\,000$; $0 \leq k_i \leq n_i$; $-1 \leq m_i \leq \min\left(\frac{n_i(n_i - 1)}{2}, 300\,000\right)$).

If $m_i = -1$, the number of trails is not fixed.

It is guaranteed that the sum of all w_i does not exceed 300 000, where $w_i = m_i$ if $m_i \neq -1$, and otherwise $w_i = \min\left(\frac{n_i(n_i - 1)}{2}, 300\,000\right)$.

Output

For each test case, output your answer in the following format:

- If a valid map exists, print `Ok` m'_i on the first line, followed by the descriptions of m'_i trails.
If $m_i \neq -1$, then $m'_i = m_i$. Otherwise, $0 \leq m'_i \leq \min\left(\frac{n_i(n_i - 1)}{2}, 300\,000\right)$.
- If no valid map exists, print `Impossible`.

If a valid map exists, each of the next m'_i lines must contain two integers u_j and v_j — the numbers of the intersections connected by the j -th trail.

It must be possible to travel between any two intersections using the trails.

Each pair of intersections can be connected by at most one trail. Each trail must connect two different intersections and can be used in both directions.

The maps do not have to be realistic in any other way.

For example, it is allowed to construct a map that cannot be drawn on a plane without trails crossing each other.

Scoring

The tests for this problem are divided into thirteen groups.

Points for a group are awarded only if all tests in that group and all tests in its required groups are passed.

Group	Points	Additional constraints	Required groups
0	0	Sample tests	–
1	8	$n_i \leq 8$ and $m_i = -1$	–
2	12	$n_i \leq 5000$ and $m_i = -1$	1
3	8	$0 \leq m_i < n_i - 1$	–
4	13	$k_i \leq 3$	–
5	6	$k_i = n_i$	–
6	10	$n_i \leq 8$	0, 1
7	10	$n_i \leq 100$	0, 1, 6
8	10	$n_i \leq 5000$ and $m_i = n_i$	–
9	4	$n_i \leq 5000$ and $m_i \geq n_i + k_i$	0
10	4	$n_i \leq 5000$	0–2, 6–9

Groups 11–13 are evaluated using **offline testing**. This means that the results of your solution on these groups will only be available after the contest ends.

Group	Points	Additional constraints	Required groups
11	5	$m_i = -1$	1, 2
12	5	$m_i = n_i$	8
13	5	–	0–12

Example

standard input	standard output
1 7 13 6	Ok 13 1 3 4 3 4 5 6 7 2 4 2 5 3 6 7 3 1 2 2 3 1 7 4 6 5 6

Note

```
FOREST(1, 0)
  dist[1] = 0
  trail 1 (1, 3): call FOREST(3, 1)
FOREST(3, 1)
  dist[3] = 1
  trail 1 (1, 3): dist[3] is not changed
  trail 2 (4, 3): call FOREST(4, 2)
FOREST(4, 2)
  dist[4] = 2
  trail 2 (4, 3): dist[4] is not changed
  trail 3 (4, 5): call FOREST(5, 3)
FOREST(5, 3)
  dist[5] = 3
  trail 3 (4, 5): dist[5] is not changed
  trail 6 (2, 5): call FOREST(2, 4)
FOREST(2, 4)
  dist[2] = 4
  trail 5 (2, 4): dist[2] = min(dist[2], dist[4] + 1) = min(4, 2 + 1) = 3
  trail 6 (2, 5): dist[2] is not changed
  trail 9 (1, 2): dist[2] = min(dist[2], dist[1] + 1) = min(3, 0 + 1) = 1
  trail 10 (2, 3): dist[2] is not changed
  trail 13 (5, 6): call FOREST(6, 4)
FOREST(6, 4)
  dist[6] = 4
  trail 4 (6, 7): call FOREST(7, 5)
FOREST(7, 5)
  dist[7] = 5
  trail 4 (6, 7): dist[7] is not changed
  trail 8 (7, 3): dist[7] = min(dist[7], dist[3] + 1) = min(5, 1 + 1) = 2
  trail 11 (1, 7): dist[7] = min(dist[7], dist[1] + 1) = min(2, 0 + 1) = 1
  trail 7 (3, 6): dist[6] = min(dist[6], dist[3] + 1) = min(4, 1 + 1) = 2
  trail 12 (4, 6): dist[6] is not changed
  trail 13 (5, 6): dist[6] is not changed
  trail 5 (2, 4): dist[4] is not changed
  trail 12 (4, 6): dist[4] is not changed
  trail 7 (3, 6): dist[3] is not changed
  trail 8 (7, 3): dist[3] is not changed
  trail 10 (2, 3): dist[3] is not changed
  trail 9 (1, 2): dist[1] is not changed
  trail 11 (1, 7): dist[1] is not changed
```

So in summary:

Crossroad	1	2	3	4	5	6	7
Distance	0	1	1	2	3	2	1

The only incorrect distance is the distance to the crossroad 5.