Treasure

A long time ago, Horus and Seth violently fought over who would succeed Osiris as King. Their contention was judged by Raa, who gave them a series of challenges to decide who was worthy of the throne. Horus managed to win all of the challenges, but Raa is still not sure if he is qualified to rule over Egypt because of his young age. So Raa decided to give Horus one final challenge to prove his strength and settle this fight once and for all.

The final challenge for Horus is to collect n treasure chests spread all over Egypt. The chests' locations are given to Horus as 2D points $(x[i], y[i])$ $(0 \leq i \leq n-1)$. Horus would then write notes on the locations in his notebook, which has pages that can each store a non-negative integer not exceeding $2 \cdot 10^9$. Sadly, Seth would not play fairly; he messes up the order of Horus's pages and shuffles them around.

Your task is to help Horus minimize the number of pages used while allowing him to successfully write and read the chests' locations regardless of how Seth decides to shuffle the pages.

Implementation details

You should implement two different procedures:

```
int[] encode(int[] x, int[] y)
```

- x, y: two arrays of length n. For each i $(0 \leq i \leq n-1)$, treasure chest i is a located at point $(x[i], y[i])$.
- This procedure should return an integer array representing the notes Horus writes in his notebook.
- Each element of the array represents a note written on a page of the notebook and must be an integer between 0 and $2 \cdot 10^9$.

```
int[] decode(int[] e)
```

- e: the integer array returned by the `encode` procedure representing Horus' notes after being shuffled by Seth.
- This procedure should return an array p of length $2n$ representing the original set of points sent to the `encode`. Each location point $(x[i], y[i])$ (for all $0 \leq i \leq n-1$) sent to `encode` should be represented by $(p[2j], p[2j+1])$ (for some $0 \leq j \leq n-1$).
- Note that the order of points returned from the `decode` doesn't have to match the order of points sent to the `encode`.

There is a total of T scenarios. For each scenario, the grader calls the `encode` procedure. The array returned is received and shuffled by the grader and then sent to the `decode` procedure. Note that in the judging system these procedures are called in separate programs.

In the first program, procedure `encode` is called once for each scenario. Invocations of procedure `decode` are made in the second program. The behavior of your implementation for each scenario must be independent of the order of the scenarios, as scenarios might not have the same order in the two programs.

Constraints

- $1 \leq T \leq 100$
- $1 \leq n \leq 4 \cdot 10^4$
- $0 \leq x[i], y[i] < 5 \cdot 10^8$ (for all $0 \leq i \leq n-1$)
- No two contestants have the same location.
- The sum of n over all the scenarios does not exceed $2 \cdot 10^5$.

Subtasks

For a given scenario t, let $k_t = l_t / n_t$ be the ratio between the number of pages Horus uses l_t and the number of treasure chests n_t. Let K be the maximum of all k_t.

1. (21 points) $0 \leq x[i], y[i] < 10^4$ (for all $0 \leq i \leq n-1$), $K \leq 4$
2. (79 points) No additional constraints. Your score for this subtask will be determined using the following rules:
    - If $K \leq 3$, you get 79 points
    - If $3 < K \leq 4$, you get 60 points
    - If $4 < K \leq 5$, you get 30 points
    - If $5 < K \leq 6$, you get 24 points
    - If $K > 6$, you get 0 points.

Example

The grader makes the following procedure call:

```
encode([1, 3, 6], [5, 2, 3])
```

In this example, there are 3 treasure chests located at points $(1, 5)$, $(3, 2)$, and $(6, 3)$. Let's assume Horus writes the numbers $[14, 18, 221, 457, 13]$ in his notebook.

Then Seth shuffles the order of the pages to make it $[457, 18, 13, 221, 14]$.

The grader then makes the following procedure call:

```
decode([457, 18, 13, 221, 14])
```

Horus examines his notes to find the original points are $(3, 2)$, $(6, 3)$, and $(1, 5)$. So the procedure returns $[3, 2, 6, 3, 1, 5]$.

Notice how, in this example, the order of points returned by the `decode` procedure is not the same as order sent to the `encode` procedure.

Sample Grader

The sample grader reads the input in the following format:

- line 1: T
- block i (for $0 \leq i \leq T - 1$): a block representing scenario i.
  - line 1: n
  - line $2 + j$ (for $0 \leq j \leq n - 1$): $x[j]$ $y[j]$

For each scenario, if the array returned by the `encode` procedure do not conform to the rules mentioned above, or the set of points returned by the `decode` is not equal to the set of points sent to the `encode`, the grader will terminate printing a line containing `WA: msg` where `msg` is the cause of the termination. Otherwise, the grader will print a single line containing `OK: l` where `l` is the length of the array returned by the `encode` procedure.