Treasure

A long time ago, Horus and Seth fought over who would succeed Osiris as King. Their contention was judged by Raa, who gave them a series of challenges to determine who is more worthy of the throne. Horus managed to win all of the challenges, but Raa is still not sure if Horus 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 numbered from 0 to $N-1$ spread all over Egypt. The locations of the chests are given to Horus as points in the 2-dimensional plane and are pairwise distinct. Specifically, the location of chest i ($0 \leq i < N$) is a point $(X[i], Y[i])$, where both $X[i]$ and $Y[i]$ are integers between 0 and $5 \cdot 10^8$, inclusive.

Horus is going to record the locations of the chests by taking notes in his papyrus notebook. Each sheet of this notebook can store a single non-negative integer not greater than $2 \cdot 10^9$. Sadly, Seth is going to shuffle the notebook sheets after Horus takes all the notes in his notebook.

Your task is to help Horus by implementing two procedures that would:

- record the locations of the chests by writing numbers on the notebook sheets,
- recover the locations of the chests, given the notebook sheets in an arbitrary order.

Note that your score in this task depends on the number of sheets used, that is, the number of numbers written in the notebook.

Implementation details

You should implement the following two procedures:

```
std::vector<int> encode(std::vector<std::pair<int, int>> P)
```

- P: array of length N specifying the location of chests. Element $P[i]$ (for $0 \leq i < N$) is a pair $(X[i], Y[i])$, denoting the location of chest i.
- This procedure should return an array E representing the numbers to be written in Horus' notebook. Each element of the array is a number written on a sheet of the notebook and must be an integer between 0 and $2 \cdot 10^9$, inclusive.
- This procedure may be called multiple times in each test case. Each call represents a different *scenario*.

```
std::vector<std::pair<int, int>> decode(std::vector<int> S)
```

- S: array of integers. This array is a shuffled version of the array returned by the `encode` procedure, i.e., it has the same length and the same elements but in a (possibly) different order.
- This procedure should return an array D of length N specifying the locations of the chests. Each element of D should be a pair (x', y') denoting the location of one chest.
- The elements in the array D, can be *in any arbitrary order*.

Let T denote the number of scenarios in a test case. The grader calls the `encode` procedure once for each scenario. Each of the T arrays returned by the `encode` procedure is then shuffled by the grader and provided to the `decode` procedure. Note that the `encode` and `decode` procedures are invoked in separate programs by the grading system. Moreover, the order of scenarios is not necessarily the same in the two programs.

Constraints

- $1 \le T \le 100$
- $1 \le N \le 40\,000$
- $0 \le X[i] \le 5 \cdot 10^8$ ($0 \le i < N$)
- $0 \le Y[i] \le 5 \cdot 10^8$ ($0 \le i < N$)
- No two chests have the same location.
- The total number of chests in *all* scenarios does not exceed $2 \cdot 10^5$.
- Array S is a permutation of E.

Subtasks and scoring

Your score in this task depends on the number of notebook sheets used, that is, the length of array E returned by the procedure `encode`. For a given scenario t, let n_t be the number of chests, and l_t be the length of array E, returned by the `encode` procedure. We define K as the maximum value of the ratio l_t / n_t among all scenarios.

Subtask	Score	Additional constraints and requirement on K
1	21	$K \le 4$; $0 \le X[i] < 10^4$ and $0 \le Y[i] < 10^4$ for each i such that $0 \le i < N$
2	79	No additional constraints.

The score for subtask 2 is computed based on K as follows:

Condition	Score
$K \leq 3$	79
$3 < K \leq 4$	60
$4 < K \leq 5$	30
$5 < K \leq 6$	24
$6 < K$	0

Example

The grader makes the following procedure call:

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

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

Then, Seth shuffles the notebook sheets to transform E to the array $S = [457, 18, 13, 221, 14]$.

The grader then makes the following procedure call:

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

Horus deduces the original points to be $(3, 2)$, $(6, 3)$, and $(1, 5)$. So the procedure returns the array $D = [(3, 2), (6, 3), (1, 5)]$. Note that the order of points returned by the `decode` procedure is not the same as the order provided to the `encode` procedure.

Sample Grader

The sample grader input starts with a line containing a single integer T followed by T scenarios. Each scenario is provided in the following format.

```
N
X[0] Y[0]
X[1] Y[1]
...
X[N-1] Y[N-1]
```

The sample grader writes the result of each of the T scenarios in the same order as they are provided in the input. The output for each scenario is as follows. Let L be the length of array E, returned by the `encode` procedure in the scenario, and M be the length of array D, returned by the `decode` procedure (M must be equal to N in a correct solution). Moreover, let $X'[i] = D[i].first$ and

$Y'[i] = D[i].second$ for $0 \leq i < M$. The sample grader prints the result of the scenario in the following format:

```
L
M
X'[0] Y'[0]
X'[1] Y'[1]
...
X'[M-1] Y'[M-1]
```

Additionally, if there are invalid elements in the array returned by `encode` in a scenario, the sample grader prints "`Invalid element in the returned array`", and terminates immediately.