Longest Trip

The IOI 2023 organizers are in big trouble! They forgot to plan the trip to Ópusztaszer for the upcoming day. But maybe it is not yet too late ...

There are N landmarks at Ópusztaszer indexed from 0 to $N-1$. Some pairs of these landmarks are connected by *bidirectional* **roads**. Each pair of landmarks is connected by at most one road. The organizers *don't know* which landmarks are connected by roads.

We say that the **density** of the road network at Ópusztaszer is **at least** δ if every 3 distinct landmarks have at least δ roads among them. In other words, for each triplet of landmarks (u, v, w) such that $0 \leq u < v < w < N$, among the pairs of landmarks $(u, v), (v, w)$ and (u, w) at least δ pairs are connected by a road.

The organizers *know* a positive integer D such that the density of the road network is at least D. Note that the value of D cannot be greater than 3.

The organizers can make **calls** to the phone dispatcher at Ópusztaszer to gather information about the road connections between certain landmarks. In each call, two nonempty arrays of landmarks $[A[0], \ldots, A[P-1]]$ and $[B[0], \ldots, B[R-1]]$ must be specified. The landmarks must be pairwise distinct, that is,

- $A[i] \neq A[j]$ for each i and j such that $0 \leq i < j < P$;
- $B[i] \neq B[j]$ for each i and j such that $0 \leq i < j < R$;
- $A[i] \neq B[j]$ for each i and j such that $0 \leq i < P$ and $0 \leq j < R$.

For each call, the dispatcher reports whether there is a road connecting a landmark from A and a landmark from B. More precisely, the dispatcher iterates over all pairs i and j such that $0 \leq i < P$ and $0 \leq j < R$. If, for any of them, the landmarks $A[i]$ and $B[j]$ are connected by a road, the dispatcher returns `true`. Otherwise, the dispatcher returns `false`.

A **trip** of length l is a sequence of *distinct* landmarks $t[0], t[1], \ldots, t[l-1]$, where for each i between 0 and $l-2$, inclusive, landmark $t[i]$ and landmark $t[i+1]$ are connected by a road. A trip of length l is called a **longest trip** if there does not exist any trip of length at least $l+1$.

Your task is to help the organizers to find a longest trip at Ópusztaszer by making calls to the dispatcher.

Implementation Details

You should implement the following procedure:

```
int[] longest_trip(int N, int D)
```

- N: the number of landmarks at Ópusztaszer.
- D: the guaranteed minimum density of the road network.
- This procedure should return an array $t = [t[0], t[1], \ldots, t[l-1]]$, representing a longest trip.
- This procedure may be called **multiple times** in each test case.

The above procedure can make calls to the following procedure:

```
bool are_connected(int[] A, int[] B)
```

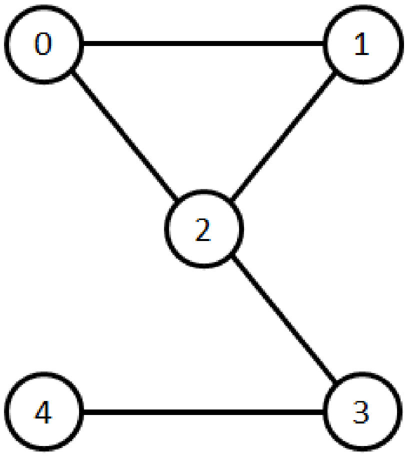
- A: a nonempty array of distinct landmarks.
- B: a nonempty array of distinct landmarks.
- A and B should be disjoint.
- This procedure returns `true` if there is a landmark from A and a landmark from B connected by a road. Otherwise, it returns `false`.
- This procedure can be called at most $32\,640$ times in each invocation of `longest_trip`, and at most $150\,000$ times in total.
- The total length of arrays A and B passed to this procedure over all of its invocations cannot exceed $1\,500\,000$.

The grader is **not adaptive**. Each submission is graded on the same set of test cases. That is, the values of N and D, as well as the pairs of landmarks connected by roads, are fixed for each call of `longest_trip` within each test case.

Examples

Example 1

Consider a scenario in which $N = 5$, $D = 1$, and the road connections are as shown in the following figure:



The procedure `longest_trip` is called in the following way:

```
longest_trip(5, 1)
```

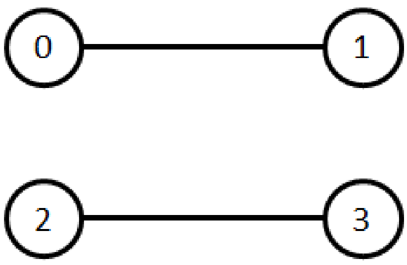
The procedure may make calls to `are_connected` as follows.

Call	Pairs connected by a road	Return value
`are_connected([0], [1, 2, 4, 3])`	$(0,1)$ and $(0,2)$	`true`
`are_connected([2], [0])`	$(2,0)$	`true`
`are_connected([2], [3])`	$(2,3)$	`true`
`are_connected([1, 0], [4, 3])`	none	`false`

After the fourth call, it turns out that *none* of the pairs $(1,4)$, $(0,4)$, $(1,3)$ and $(0,3)$ is connected by a road. As the density of the network is at least $D = 1$, we see that from the triplet $(0,3,4)$, the pair $(3,4)$ must be connected by a road. Similarly to this, landmarks 0 and 1 must be connected.

At this point, it can be concluded that $t = [1,0,2,3,4]$ is a trip of length 5, and that there does not exist a trip of length greater than 5. Therefore, the procedure `longest_trip` may return $[1,0,2,3,4]$.

Consider another scenario in which $N = 4$, $D = 1$, and the roads between the landmarks are as shown in the following figure:



The procedure `longest_trip` is called in the following way:

```
longest_trip(4, 1)
```

In this scenario, the length of a longest trip is 2. Therefore, after a few calls to procedure `are_connected`, the procedure `longest_trip` may return one of $[0, 1]$, $[1, 0]$, $[2, 3]$ or $[3, 2]$.

Example 2

Subtask 0 contains an additional example test case with $N = 256$ landmarks. This test case is included in the attachment package that you can download from the contest system.

Constraints

- $3 \le N \le 256$
- The sum of N over all calls to `longest_trip` does not exceed $1\,024$ in each test case.
- $1 \le D \le 3$

Subtasks

1. (5 points) $D = 3$
2. (10 points) $D = 2$
3. (25 points) $D = 1$. Let $l^\star$ denote the length of a longest trip. Procedure `longest_trip` does not have to return a trip of length $l^\star$. Instead, it should return a trip of length at least $\left\lceil \frac{l^\star}{2} \right\rceil$.
4. (60 points) $D = 1$

In subtask 4 your score is determined based on the number of calls to procedure `are_connected` over a single invocation of `longest_trip`. Let q be the maximum number of calls among all invocations of `longest_trip` over every test case of the subtask. Your score for this subtask is calculated according to the following table:

Condition	Points
$2\,750 < q \le 32\,640$	20
$550 < q \le 2\,750$	30
$400 < q \le 550$	45
$q \le 400$	60

If, in any of the test cases, the calls to the procedure `are_connected` do not conform to the constraints described in Implementation Details, or the array returned by `longest_trip` is incorrect, the score of your solution for that subtask will be 0.

Sample Grader

Let C denote the number of scenarios, that is, the number of calls to `longest_trip`. The sample grader reads the input in the following format:

- line 1: C

The descriptions of C scenarios follow.

The sample grader reads the description of each scenario in the following format:

- line 1: $N \ D$
- line $1 + i$ ($1 \leq i < N$): $U_i[0] \ U_i[1] \ \ldots \ U_i[i-1]$

Here, each U_i ($1 \leq i < N$) is an array of size i, describing which pairs of landmarks are connected by a road. For each i and j such that $1 \leq i < N$ and $0 \leq j < i$:

- if landmarks j and i are connected by a road, then the value of $U_i[j]$ should be 1;
- if there is no road connecting landmarks j and i, then the value of $U_i[j]$ should be 0.

In each scenario, before calling `longest_trip`, the sample grader checks whether the density of the road network is at least D. If this condition is not met, it prints the message `Insufficient Density` and terminates.

If the sample grader detects a protocol violation, the output of the sample grader is `Protocol Violation: <MSG>`, where `<MSG>` is one of the following error messages:

- `invalid array`: in a call to `are_connected`, at least one of arrays A and B
  - is empty, or
  - contains an element that is not an integer between 0 and $N-1$, inclusive, or
  - contains the same element at least twice.
- `non-disjoint arrays`: in a call to `are_connected`, arrays A and B are not disjoint.
- `too many calls`: the number of calls made to `are_connected` exceeds $32\,640$ over the current invocation of `longest trip`, or exceeds $150\,000$ in total.
- `too many elements`: the total number of landmarks passed to `are_connected` over all calls exceeds $1\,500\,000$.

Otherwise, let the elements of the array returned by `longest_trip` in a scenario be $t[0], t[1], \ldots, t[l-1]$ for some nonnegative l. The sample grader prints three lines for this scenario in the following format:

- line 1: l
- line 2: $t[0] \ t[1] \ \ldots \ t[l-1]$
- line 3: the number of calls to `are_connected` over this scenario

Finally, the sample grader prints:

- line $1 + 3 \cdot C$: the maximum number of calls to `are_connected` over all calls to `longest_trip`