Kevin and Teams

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	256 megabytes

This is an interactive problem.

Kevin has n classmates, numbered $1, 2, \ldots, n$. Any two of them may either be friends or not friends.

Kevin wants to select $2k$ classmates to form k teams, where each team contains exactly 2 people. Each person can belong to at most one team.

Let u_i and v_i be two people in the i-th team. To avoid potential conflicts during team formation, the team members must satisfy one of the following two conditions:

- For all i $(1 \le i \le k)$, classmate u_i and v_i are friends.
- For all i $(1 \le i \le k)$, classmate u_i and v_i are not friends.

Kevin wants to determine the maximum k such that, regardless of the friendship relationships among the n people, he can always find $2k$ people to form the teams. After that, he needs to form k teams. But asking whether two classmates are friends is awkward, so Kevin wants to achieve this while asking about the friendship status of no more than n pairs of classmates.

The interactor is **adaptive**. It means that the hidden relationship between classmates is not fixed before the interaction and will change during the interaction.

Interaction Protocol

Each test contains multiple test cases. The first line contains the number of test cases t $(1 \le t \le 10^4)$. The description of the test cases follows.

The first line of each test case contains one positive integer n $(2 \le n \le 10^5)$ — the number of classmates.

First, you should output an integer k $(1 \le k \le \frac{n}{2})$ — the maximum number of teams you can form.

You can make queries in the following way — print one line of the form ? u v where $1 \le u \ne v \le n$. After that, read a single integer: 0 or 1 indicating whether they are friends. 1 means they are friends while 0 means not.

If you want to print the answer, output ! u_1 v_1 u_2 v_2 ... u_k v_k. You should output **exactly** $2k$ distinct numbers. Then, the interaction continues with the next test case.

You can make at most n queries. Printing the answer does **not** count towards the number of queries made.

It is guaranteed that the sum of n over all test cases does not exceed 10^5.

After printing each query do not forget to output the end of line and flush* the output. Otherwise, you will get `Idleness limit exceeded` verdict.

If, at any interaction step, you read -1 instead of valid data, your solution must exit immediately. This means that your solution will reveive `Wrong answer` because of an invalid query or any other mistake. Failing to exit can result in an arbitrary verdict because your solution will continue to read from a closed stream.

Hacks

*To flush, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `sys.stdout.flush()` in Python;
- see the documentation for other languages.

The interactor for hacks is not adaptive. To make a hack, use the following format.

The first line contains the word "`manual`".

The second line contains a single integer t ($1 \le t \le 10^4$) — the number of test cases.

The first line of each test case contains an integer n ($2 \le n \le 10^5$) — the number of classmates.

The second line of each test case contains a string s consisting of '0' and '1' of length $\frac{n(n-1)}{2}$ — indicating the relationship between $(1,2)$, $(1,3)$, ..., $(1,n)$, $(2,3)$, $(2,4)$, ..., $(2,n)$, ..., $(n-1,n)$. '1' means being friends and '0' means not being friends.

The following is the input of example.

```
manual
2
3
011
5
1011101011
```

The sum of n over all test cases must not exceed 10^5.

There is an additional constraint for hacks: The sum of $\frac{n(n-1)}{2}$ over all test cases must not exceed 10^7.

Example

standard input	standard output
2 3 1 5 1 0 1 0 0	 1 ? 1 2 ! 1 2 2 ? 1 2 ? 3 4 ? 3 5 ? 1 3 ? 2 4 ! 1 2 3 5

Note

In the first test case:

Kevin claims he can form 1 team regardless of the friendship relationships among the 3 people.

Kevin asks about the friendship relationship between people 1 and 2. The jury responds that they are friends.

Kevin answers that he can form a team with people 1 and 2.

In the second test case:

Kevin claims he can form 2 teams regardless of the friendship relationships among the 5 people.

Kevin asks about the friendship relationship between people $(1, 2), (3, 4), (3, 5), (1, 3), (2, 4)$. The jury responds with $1, 0, 1, 0, 0$.

Kevin answers that he can form two teams with people $(1, 2)$ and $(3, 5)$.

It is also possible to form two teams with people $(1, 3)$ and $(2, 4)$, since they are both not friends.