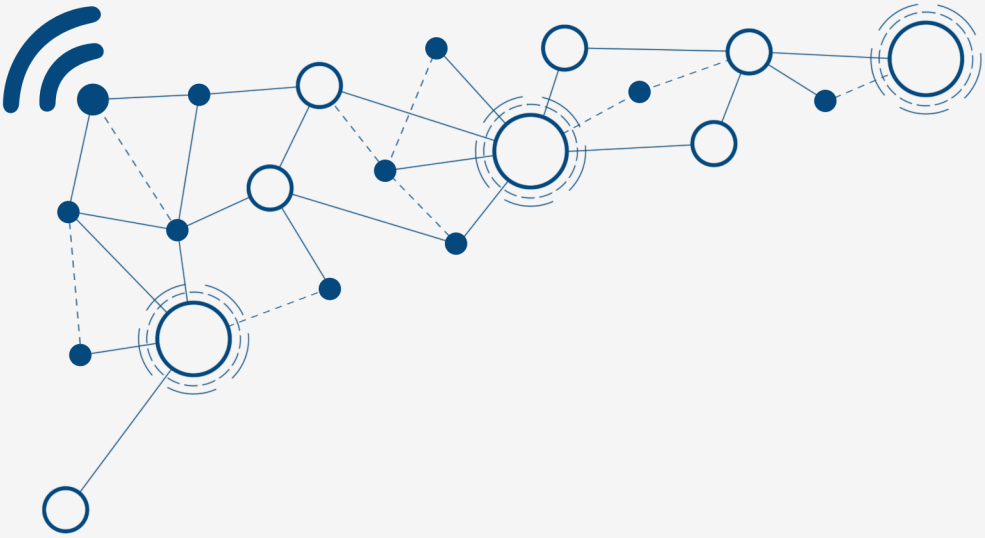
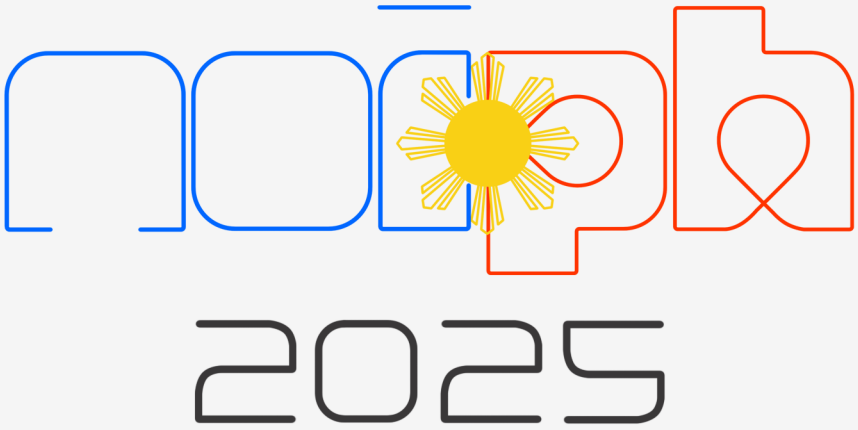
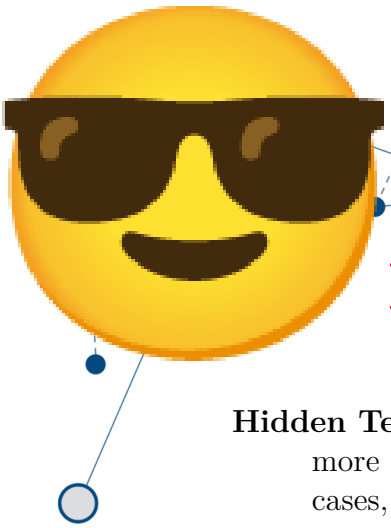




National Olympiad in Informatics

Finals Round 1



Important! Read the following:

Hidden Test Cases. Your solution will be checked by running it against one or more (usually several) hidden test cases. You will not have access to these cases, but a correct solution is expected to handle them correctly.

Strict Output Format. The output checker is **strict**. Follow these guidelines strictly:

- It is **space sensitive**. Do not output extra leading or trailing spaces. Do not output extra blank lines unless explicitly stated.
- It is **case sensitive**. So, for example, if the problem asks for the output in lowercase, follow it.
- Do not print any tabs. (No tabs will be required in the output.)
- Do not output anything else aside from what's asked for in the Output section. So, do not print things like "`Please enter t`".

Not following the output format strictly and exactly will likely result in the verdict "*Output isn't correct*".

Use Standard I/O. Do not read from, or write to, a file. You must read from the standard input and write to the standard output.

Submit Code Only. Only include **one** file when submitting: the source code (.cpp, .py, etc.) and nothing else.

No Java Package. For Java submissions, do not include a `package` line.

No Weird Filenames. Only use letters, digits and underscores in your filename. Do not use spaces or other special symbols.

Use Fast I/O. Many problems have large input file sizes, so use fast I/O. For example:

- In C/C++, use `scanf` and `printf`.
- In Python, use `sys.stdin.readline()`

Flush On Interactive Problems. On interactive problems, make sure to **flush** your output stream after printing.

- In C++, use `fflush(stdout);` or `cout << endl;`
- In Python, use `sys.stdout.flush()` or `print(flush=True)`
- For more details, including for other languages, ask a question/clarification through CMS.

Good luck and enjoy the contest! 😃

Contents

Notes

- Many problems have large input file sizes, so use fast I/O. For example:
  - In C/C++, use `scanf` and `printf`.
  - In Python, use `sys.stdin.readline()`
- On interactive problems, make sure to **flush** your output stream after printing.
  - In C++, use `fflush(stdout);` or `cout << endl;`
  - In Python, use `sys.stdout.flush()` or `print(flush=True)`
  - For more details, including for other languages, ask a question/clarification through CMS.

Good luck and enjoy the problems!

Problem C
Electra Boom

We're NOI.PH guys, of course we think ***interactive*** *is the best type of problem!*

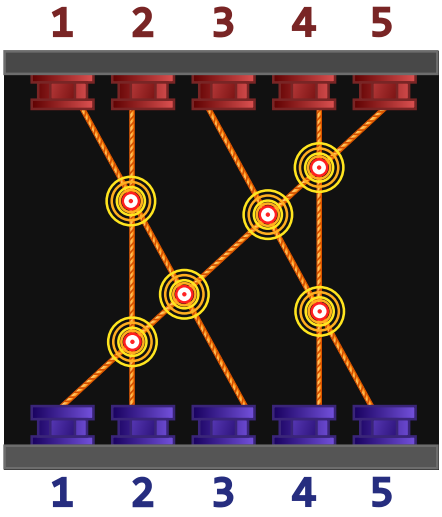
Electra Boom is one of my favorite YouTubers! She works with circuitry and electronics, and she makes it approachable by being unafraid to show things going wrong (and usually when they go wrong, they go *BOOM!*).

In one of her series, she buys shady electronics parts on the internet and tests them to see how badly they malfunction. The more catastrophic the failure, the higher she ranks it on the *B💣OM METER*!

For this week's episode, she has acquired an ominous-looking blackbox from the website Teμ. This black box has n **input nodes** on top, and n **output nodes** at the bottom, each arranged in a row and labeled 1 to n from left to right. There are n wires of negligible width, connecting each of the input nodes with a different output node.

In a somewhat-concerning design choice, the wires are bare and have no form of insulation whatsoever. Also, the wires are pressed flat together when the blackbox is shut. This means that if two wires have to cross over each other in some configuration, then there *will* be a point where those two wires make direct contact! At the very least, due to how the nodes were spaced apart, it is given that it is impossible for three wires to ever meet at a single point.

For an example, consider the configuration illustrated in the following diagram. The points of contact have all been highlighted.



If Electra Boom runs a live current through this blackbox, each of those points of contact makes a *BOOM!* Precisely, she hears one *BOOM* for every pair of wires that make contact in the blackbox's current configuration (and she is able to precisely count the number of *BOOMs* by performing a fourier transform on the waveform of the audio), and this number serves as the product's ranking on the *B💣OM METER*.

This week's video is special... because it's not a video, it's a livestream! She's going to be reading chat for suggestions on how she should tinker with the blackbox.

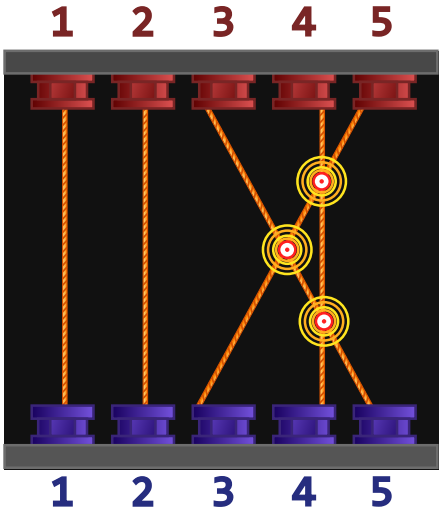
Due to the sealed nature of the blackbox, this is the only operation available to Electra Boom:

- Choose two output nodes i and j, and swap the wires connected to those endpoints.
    - Formally, the wire that was connected to output node j is now connected to output node i, and vice versa.

It is guaranteed that the wires will not "snag" on each other after each swap—that is, each wire always remains a straight line connecting its two endpoints.

Electra Boom (irresponsibly) runs live current through the blackbox, and reports to her audience the blackbox's rating on the *B💣OM METER*. She also (still irresponsibly) runs live current through the blackbox after *each* operation, and then reports the rating on the *B💣OM METER* after each operation.

For example, in the diagram from earlier, Electra Boom would report the blackbox as getting a 6 on the *B💣OM METER*. After swapping the wires connected to output nodes 1 and 3, it looks like this. Electra Boom would now report that the blackbox gets a 3 on the *B💣OM METER*.



It's your chance to keep Electra Boom safe by bringing the blackbox's rating on the *B💣OM METER* down to 0. And for the sake of her wellbeing, please try your best to do so in as few operations as you can manage.

We're NOI.PH guys, of course we like problem statements with nonsensical stories!

We're NOI.PH guys, of course we enjoy squeezing points out of problems with special scoring formulas!

Interaction

The interactor first sends an integer T, the number of test cases. Each test case proceeds as follows.

The interactor first sends a line containing the two space-separated integers n and I—the number of nodes, and the blackbox's initial rating on the *B💣OM METER* when it is first plugged in.

To tell Electra Boom what operation to do, print a line containing two space-separated integers i and j (where $1 \le i, j \le n$, and $i \neq j$), denoting the output node endpoints whose wires you want to swap.

When printing this line, **be sure to flush the output** so that Electra Boom can see your comment in the chat. Otherwise, Electra Boom might not receive your output, and she will wait forever.

- In C++, use `fflush(stdout);` to flush the stream, or `cout << endl;` to print a newline character and then flush the stream.
- In Python, use `sys.stdout.flush()` to flush the stream, or use `print(flush=True)` to print a newline character and then flush the stream.
- For more details, including for other languages, ask a question/clarification through CMS.

After that, the interactor responds by sending a line containing a single integer B.

- If $B = -1$, this means that you have either made an invalid output or exceeded the number of allowed operations. End the interaction to obtain a Wrong Answer verdict.
- Otherwise, B is the number of *BOOMs* that Electra Boom heard after performing your proposed operation.
  - In particular, if $B = 0$, then the blackbox has been neutralized (i.e. no *BOOM*s were heard). **When this happens, the next test case immediately begins.**
  - On the other hand, if $B > 0$, suggest another operation.

You can make at most 10000 operations in a single test case.

Constraints

For all subtasks
$1 \leq T \leq 20$ $I > 0$

Subtask	Points	Constraints
1	**15**	$n \leq 7$
2	**25**	$n \leq 99$
3	**60**	$n \leq 300$

There is also partial scoring for subtasks 2 and 3.

For each subtask, if you got a -1 response in any test case in any test file under that subtask, you get 0 points for that subtask.

Otherwise, let Q be the maximum number of queries made in any single test case, across all test files under that subtask.

- If you pass all test files under subtask 1, your score is 15.
- If you pass all test files under subtask 2, your score is $\min\left(25,\ 25 \cdot \left(\frac{5000}{Q}\right)\right)$.
- If you pass all test files under subtask 3, your score is $\min\left(60,\ 12 \cdot \log_2\left(\frac{20000}{Q}\right)\right)$.

Sample Interaction

The blank lines have only been added to illustrate the chronology of the back-and-forth between the submission and the interactor.

```
Input       Output
2
3 1
            1 2
2
            2 3
3
            3 1
0
2 1
            1 2
0
```

Testing Tool

A testing tool written in Python 3 is provided in the attachments in CMS to help with local testing.

- `interactive_runner.py` is a special program that pipes the input/output of your solution with the output/input of the interactor.
- `testing_tool.py` serves as the interactor, processing the logic that the judge is responsible for.
- `input.txt` is where we store the values of the parameters (the "input" of the interactor) that we would like to test our solution against. In this case, it contains the initial configuration of the blackbox.

The format of "`input.txt`" is as follows. A sample `input.txt` file is also included as an attachment.

The first line of the file contains a single integer T. This should be followed by the descriptions of the T test cases.

The first line of each test case should contain the integer n. Since there are n wires, suppose we arbitrarily label those wires $1, 2, 3, \ldots, n$.

The second line of each test case should contain n space-separated integers, where the ith integer is the label of the wire connected to input node i. No label should appear more than once.

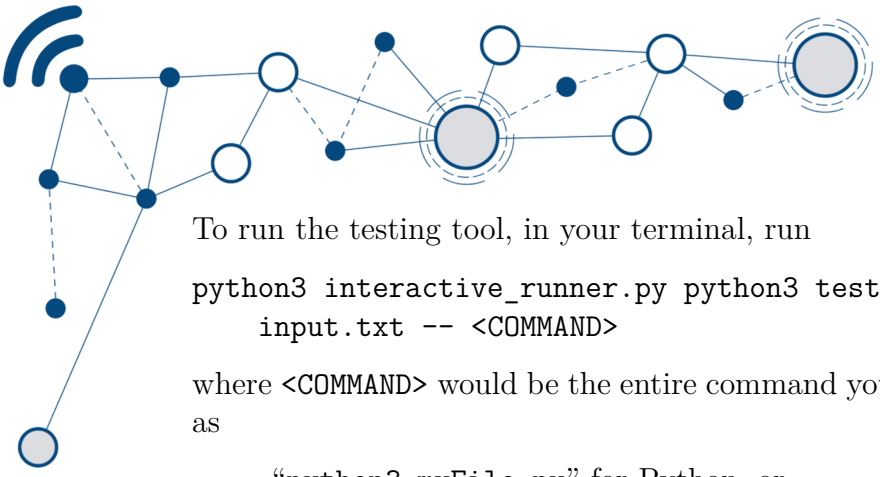
Similarly, the third line of each test case should also contain n space-separated integers, where the ith integer is the label of the wire connected to *output* node i. No label should appear more than once.

For example, the test case

```
5
3 1 2 5 4
4 1 3 5 2
```

in `input.txt` would mean:

- Input node 2 and output node 2 are connected by the wire labeled 1
- Input node 3 and output node 5 are connected by the wire labeled 2
- Input node 1 and output node 3 are connected by the wire labeled 3
- Input node 5 and output node 1 are connected by the wire labeled 4
- Input node 4 and output node 4 are connected by the wire labeled 5



To run the testing tool, in your terminal, run

```
python3 interactive_runner.py python3 testing_tool.py
    input.txt -- <COMMAND>
```

where `<COMMAND>` would be the entire command you use to run your solution, such as

- "`python3 myFile.py`" for Python, or
- "`./executable`" for C++ (note that this is the compiled program, not the source code)

For more verbose output (which might help in debugging), you may add an integer from 0 to 2 before the `--` in the command (where 0 is the default, and 2 is maximum verbosity).

So, for example,

```
python3 interactive_runner.py python3 testing_tool.py
    input.txt 1 -- python3 myFile.py
```

If your solution would be marked as Wrong Answer, the testing tool will output an appropriate message. Otherwise, it will output "`CORRECT`".