This is an optional tutorial file in case you are new to Interactive Problems, and/or you want to test your solution in depth. Note that it may take you 3-5 minutes to set up.

1. Example Files

The following archive contains example solution files found in the directory `examples`. Each file has the logic of each query implemented, so that you only have to focus on the guessing strategy.

We have provided implementations in all supported programming languages.

2. Interaction Script

2.1 Interactors

Normally, you do not need to execute Interactors manually. See the next section for how to start an interaction.

We have made interactors in Bash as well in all supported programming languages:

Interactor	Language	Source	Operating System
`bin/interactor.sh`	Bash	Same File	Linux
`bin/interactor.py`	Python	Same File	Windows, Linux
`bin/cinteractor`	C	`src/interactor.c`	Linux
`bin/cxxinteractor`	C++	`src/interactor.cpp`	Linux
`bin/interactor.jar`	Java	`src/interactor.java`	Windows, Linux
`bin/cinteractor.exe`	C	`src/interactor.c`	Windows
`bin/cxxinteractor.exe`	C++	`src/interactor.cpp`	Windows

> For C and C++. We have provided binaries for **both** Linux and Windows with `x86_64` architecture.
>
> If the binary does not work under your platform, you can regenerate the desired interactor by recompiling the source code under the directory `src`.

2.2 Interaction

To quickly setup interaction: See the example corresponding to your Programming Language.

To test your solution, you have to compile your application (if applicable) and start the interaction.

2.2.1 Windows

```
python ./start_interaction.py TEST_FILE @ INTERACTOR  ARGS... @ EXECUTABLE
ARGS...
```

2.2.2 Linux

In Linux, you **can** use the command provided in the Windows section, or use:

```
./start_interaction.sh TEST_FILE @ INTERACTOR  ARGS... @ EXECUTABLE ARGS...
```

2.2.3 Arguments

In both cases, the arguments are as follows:

- TEST_FILE is the name of the test file, examples are found in the `example` directory. If you want to read from standard input, put dash: `-`.
- *INTERACTOR* is the name of the interactor. We **recommend** `./bin/interactor.py` for simplicity. Otherwise, choose the interactor matching your preference. The interactor and your solution **do not have** to be in the same programming language.
- *EXECUTABLE* is the name of your application.

2.3 Programming Languages

The *EXECUTABLE* variable depend on **your solution's** programming language as follows:

Solution Language	Interaction	Notes
C / C++	- *EXECUTABLE* should be the path of the solution - ARGS... should be empty	You should first compile your solution to an executable.
Java	- *EXECUTABLE* should be `java` - ARGS... should be equal to the path of the java class	You should first compile your solution to a java class.
Python	- *EXECUTABLE* should be `python` - ARGS... should be equal to the path of the java class	

2.4 What does it do?

The script will simulate the interaction process, and output the verdict. It will also **generate** a file called interaction.txt that details the interaction process. As an example:

Problem: Find a hidden array P of strictly positive integers given its size N

- Query: `? i j`. Query elements i and j.
- Response: `P[i] × P[j]`. Get the product between P_i and P_j
- Answer: `! P1 ... PN`. Guess the array P

Example interaction with N=4.
Content of ***interaction.txt***:

```
Received Input: ? 1 2
Sent Output: 1
Received Input: ? 1 3
Sent Output: 5
Received Input: ? 1 4
Sent Output: 4
Received Input: ! 1 1 5 3
Wrong answer!
Expected: 1 1 5 4
Found:    1 1 5 3
```

If there is an unexpected error in the `start_interaction.sh` script. Please send a clarification to the Judges.

3. Examples

Each section shows how to start an interaction depending in the programming language of your solution.

3.1 C/C++ Solutions

1. Start executable named `solution` (found in the same directory) and read it from standard input using Bash interactor

```
./start_interaction.sh - @ ./bin/interactor.sh @ ./solution
```

2. Start executable named `solution` (found in the same directory) and read file `sample/02.in` using Python interactor

```
./start_interaction.sh sample/02.in @ python ./bin/interactor.py @
./solution
```

3.2 Python Solutions

1. Start python solution and read file `sample/01.in` using `cinteractor`

```
./start_interaction.sh sample/01.in @ ./bin/cinteractor @ python example.py
```

2. Start python solution named `solution.py` (found in the same directory) and read it from standard input using Python interactor

```
./start_interaction.sh - @ python ./bin/interactor.py @ python solution.py
```

3.3 Java

1. Start a compiled Java solution with the name `solution.class` and read standard input. Interactor is `./bin/cxxinteractor`

```
./start_interaction.sh - @ ./bin/cxxinteractor @ java solution
```

2. Start a compiled Java solution with the name `solution.class` and read standard input. Interactor is `interactor.jar`

```
./start_interaction.sh - @ java -jar ./bin/interactor.jar  @ java solution
```