Frequently Asked Questions

Please carefully review the Coding Competitions Terms and the Code Jam Rules first, as many of your questions regarding eligibility, contest structure, prizes, grounds for disqualification, and more are addressed there.

Expand All

Overview

Registering

Preparing

Competing

Coding

Platform

Testing

Interactive Problems

What is an interactive problem?

In an interactive problem, our judging system and your code run at the same time. When you send output to your output stream, the judge reads that output and then responds by sending input to your input stream. Then your code reads that input, and determines what output to send next, and so on. Each test set proceeds in this way. Generally, the judge tells your code when a test set has been solved, or when you have failed the test set.

For example, one classic interactive problem that you may have seen in real life is a number guessing game in which you have a certain number of tries to figure out a number that a friend has chosen. You make a guess, and your friend tells you whether the guess is too high, too low, or correct. If you are not correct, then, based on that information, you can make another guess, and get a response; this continues until you either guess the right number, run out of tries, or make an invalid guess like a non-number that causes your friend to get mad and stop playing.

How do interactive problems work?

Just as in non-interactive problems, you will receive input from stdin and print output to stdout. Our system will do the job of directing your output stream to the judge's input stream, and directing the judge's output stream to your input stream. However, every time you output data, **you must flush your output buffer**, as explained below.

Anything your program sends through standard error is ignored, but it might consume some memory and be counted against your memory limit, so do not overflow it.

If your program continues to wait for the judge after receiving an indication that your answer is incorrect (as described in the problem statement), your program will time out, resulting in a Time Limit Exceeded error. Notice that it is your responsibility to have your program exit in time to receive a Wrong Answer judgment instead of a Time Limit Exceeded error. As usual, if the total time or memory is exceeded, or your program gets a runtime error, you will receive the appropriate judgment.

You should not send additional information to the judge after processing all test cases. In other words, if your program keeps printing to standard output after the last test case, you will get a Wrong Answer judgment.

What does it mean to flush a buffer?

Many programming languages choose to "buffer" the output of your program. That is, when you send output, instead of being printed immediately to the output stream, it may be stored in a buffer. That entire buffer is then printed to the output stream at some later time (usually when your code terminates).

Buffering is a great language feature in most situations, but it interferes with interactive problems! If you print output without flushing, and the output gets buffered, the judge will wait forever for output that never comes, and so your code will also wait forever for a judge response that never comes, causing a deadlock.

You will need to manually flush the buffer every time that you print output, in order for the judging system to see it. Different languages have different syntax for output flushing. You can learn more by consulting the specification of the programming language of your choice.

Is there a sample interactive solution in my favorite language?

The analysis for the Number Guessing problem from the Code Jam 2018 Practice Session includes sample solutions in several languages. Check it out! We hope that the overall pattern can easily be adapted to any language of your choice.

How can I test an interactive solution?

You will need to be able to run Python 3 locally.

to behave exactly the same way as the ones that will evaluate your problem – in particular, they may implement randomness in a different way. However, they should help you test your solution.

To test your solution locally, use our interactive_runner script in conjunction with the judge. Instructions are included in the comments.

To test your solution on our server, toggle "Test run mode" on the bottom of the problem page. Add the judge to the bottom pane and select the appropriate language (generally Python 3). Also modify the judge to include the test cases of your choice. Then add your solution to the top pane, make sure to select a language for your code (it may be a different language than the judge's), and click "Run tests".

When running remotely, the judge will be passed a 0 as a command line argument. If you are using one of our judges, this means the judge will run your code against whatever cases you have specified for the first test set.

Be aware that the system will not check the validity of your test cases or judge against any limits or specifications.

Remember that there is a 20 second time limit for each test run, regardless of the time limits stated in the problem, so you may need to limit the data size and the number of cases you add for each run.

For practicing with problems from Code Jam 2018 or 2019, you can use the local testing tool included with each problem.

Submitting

Scoring

Communicating

Social Media

Still have a question?

If you have a question which is not answered in the above FAQ, you can email us at codejam@google.com. Please note that we may need some time to answer your question. Thanks for your understanding!

Looking for more challenges?

Code Jam is one of the three competitions that Google holds for participants of all skill levels. Find your challenge.

View other competitions