Advice

Look at the pretests. You can access the first four pretests for each problem once you've made a submission. Some of the pretests are reduced in size to help you debug your program. Keep in mind that your final submission will be judged on a separate set of 40 hidden system tests for the official rankings.

Understand the new scoring system. Your program will only be submitted to the 40 system tests once it passes all 10 pretests. You will get one point for each system test you pass, **plus 20 points** if you get all of them correct (for a maximum of 60 points per problem). Results of the system tests will not be released until the end of the contest. Unlike last year, you will not get any points for programs that do not pass all pretests. Ties will be broken by the time of the last submission of a program which passes pretests.

Watch out for integer overflow. When a problem uses large values, make sure you use `long` (in Java) or `long long` (in C++). Python integers cannot overflow.

Use fast I/O. For problems with large input sizes, you may want to use faster I/O methods to prevent a time limit error. Here is how to use fast I/O in each language:

- In Python, write `from sys import stdin, stdout` at the top of your program. When reading input, use `stdin.readline()`. To write output, use `stdout.write()`.
- In Java, use a custom Scanner class as shown here.
- In C++, write `ios_base::sync_with_stdio(false); cin.tie(NULL);` at the top of your `main` method. Then you can use `cin` and `cout` as usual. Printing a single newline character (`\n`) is faster than `endl`.

Print extra digits for non-integer values in C++. If you are printing a `double` value in C++, by default it will only output a few digits (which may result in a wrong answer from our grader). To output real values with more precision, write `cout << setprecision(16);` at the top of your program. Our grader will accept real values in fixed **or** scientific notation (so `1234.56789`, `1.23456789E3`, and `1.23456789e+003` are treated the same). There will always be a tolerance for small relative errors between your solution and the correct answer.

Special considerations for Python. Besides using fast I/O, here are some ways you can speed up your Python solutions. First, use array flattening instead of nested lists. You can increase the recursion limit if your functions uses repeated recursion. Additionally, we strongly recommend submitting your solutions in **PyPy**, which is typically faster than Python. Finally, `exit()` does not work with our grader.

Language versions. Our grader uses C++17 (compiled with the `-O3` tag), Java 11, and Python 3.9. Our version of PyPy implements Python 3.7.

Ask for clarifications! If you are confused about a problem statement, do not hesitate to message us.