ICPC North America Regionals

icpc international collegiate programming contest

The 2021 ICPC Mid-Atlantic
USA Regional Contest

Contest Guide

Welcome to the 2021 ICPC Mid-Atlantic Regional.
This document outlines the rules and procedures that will be in effect for the contest.

The Contest

1. The contest web interface is at `https://mausa21.kattis.com`.

   If you notice that the web page does not show correct timestamp or time zone, open "Settings" from the menu at the top right, and then update "Preferred time zone".

2. There will be one or more problems for the practice round. For the actual contest, there will be between 8 to 15 problems.

   The problems may appear in any order. **They are not necessarily sorted by difficulty**.

3. The winning team is the one that successfully completes the most problems in the time allowed. All problems are weighted equally, regardless of their difficulties.

   If teams are tied with the same number of problems solved, the tie is broken in favor of the team with the fewest penalty points. For each problem *solved correctly*, penalty points are charged as the sum of

   - the number of minutes elapsed since the start of the contest to when the successful submission was made, and
   - 20 points for each incorrect submission prior to the successful one.

   No penalty points are added for problems that are never solved.

4. Although Python and Kotlin are accepted as programming languages in this contest, no guarantee is made that a Python or Kotlin solution is possible that runs within the time limits allowed for any given problem.

Submitting Your Programs

1. All important submission requirements can be found at
   `https://mausa21.kattis.com/help`.
   In particular, for how to submit your programs, see
   `https://mausa21.kattis.com/help/submit`.

2. You will submit problems via the web interface. Log in to the web interface and choose "submit". You will be presented with a web form where you can upload files or enter the source code.

3. Your program must be contained within a single source-code file. Java programs should be written in the default (unnamed) package, meaning that it should not contain a `package` statement at all.

4. **All solutions must read from standard input and write to standard output**.

   Output sent to the standard error stream will be ignored and will not affect the judging of whether your output is correct. You are not obligated, therefore, to remove debugging output printed to standard error. (Such output does, however, take time and, if excessive, could cause your program to be rejected due to a Time Limit Exceeded error.)

5. All input sets used by the judges will follow the input format specification found in the problem description. You do not need to test for input that violates the input format specified in the problem.

6. All lines of program input will end with the appropriate line terminator (e.g., a linefeed on Unix/Linux systems, a carriage return-linefeed pair on Windows systems).

7. Unless otherwise specified, all lines of program output

   - must be left justified, with no leading blank spaces prior to the first non-blank character on that line,
   - must end with the appropriate line terminator (`\n`, `endl`, or `println()`), and
   - must not contain any blank characters at the end of the lines, between the final specified output and the line terminator.

   You must not print extra lines of output, even if empty, that are not specifically required by the problem statement.

8. If a problem calls for floating-point output, it will specify a *tolerance* in the output description. Your output will be considered correct if it has an absolute or relative error less than the tolerance, whichever is larger.

   For example, if the correct answer is 12300.0 and the problem gives a tolerance of $.01$, then your output would be accepted it if falls within ± 123.0 of the correct answer, as that falls within a relative error of ± 0.01.

   On the other hand, if the correct answer is 0.05 and the problem gives a tolerance of $.01$, then your output would be accepted it if falls within ± 0.01 of the correct answer, as that falls within an absolute error of ± 0.01.

9. For compiler settings, see `https://mausa21.kattis.com/help`. Some unexpected differences in environments may exist. It is, in general, your responsibility to write your code in a portable manner compliant with the rules and standards of the programming language. You should not rely upon undocumented, non-standard, or deprecated behaviors.

10. The submission of code deliberately designed to delay, crash, or otherwise negatively affect the contest itself will be considered grounds for immediate disqualification.

Judging

1. Solutions for problems submitted for judging are called runs. Each run will be judged. This judging may be entirely automated or may be managed by a human.

2. The judges will execute your program on multiple test cases. To be successful, each test execution must return the correct output, formatted as specified in the problem statement, and must complete execution within the appropriate time limit.

3. All problems will have a time limit specified at the top of the problem statement. The time limit may also be found on the web interface.

   You cannot know for sure whether your own processor is faster or slower than the judging hardware, nor by how much this might be true. Coping with this uncertainty is considered to be part of the challenge in the contest, so you should generally attempt to design solutions that beat the time limit by a healthy margin.

4. The judges will respond to your submission with one of the verdicts listed at `https://mausa21.kattis.com/help/judgements`.

   Additional notes on these verdicts:

   - Test cases are processed one by one until all have been completed or a verdict other than "Accepted" is indicated. It is possible, therefore, that a program would have generated "Wrong Answer", "Run Time Error", and "Time Limit Exceeded" on different test cases, in which case you will only receive the first such verdict encountered that may not reflect all the issues your solution might have.
   - A "Compile Error" can be issued if, during submission, you give the wrong information about the main Java class, or about the programming language (including using an incorrect file extension).
   - A "Wrong Answer", "Time/Memory Limit Exceeded", or "Runtime Error" verdict can be generated if, during submission, you mis-identify which problem your source code tries to solve.
   - Make sure you return 0 from your C/C++ `main` functions or you may get a "Run Time Error".

     In Java, uncaught exceptions or termination with a non-zero exit status will be judged as Run Time Errors.

5. You can track the progress of your submissions by logging into the web interface and choosing your name from the top right menu. On this page you will see a list of all submissions you have made, in reverse chronological order. As the submission proceeds through the judgement process your submissions page will reflect this.

Clarifications

1. In the event that you feel a problem statement is ambiguous or incorrect, you may request a clarification. Read the problem **carefully** before requesting a clarification.

   If a clarification is issued during the contest, it may be either replied to your team privately, or broadcast to all teams.

   If the judges believe that the problem statement is sufficiently clear, you will receive the response: "*No comment, read problem statement.*" If you receive this response, you should read the problem description more carefully. If you still feel there is an ambiguity, you will have to be more specific or descriptive of the ambiguity you may have found.

2. To issue a clarification request, click on Clarifications in the contest web interface.

   The clarifications page has three sections:

   - submitted clarification requests from your team that have not yet been answered,
   - a form for submitting a clarification request, and
   - clarification requests with answers from the judges.

   When you submit a clarification request, please select a subject (either one of the problems or "general") and write your request *clearly*.

3. By the day of the the contest, the problem descriptions have been reviewed carefully by many different people. So if you submit a clarification that simply says "Problem A is ambiguous." or "The sample output for problem B is incorrect.", the judges will have no particular reason to believe you and will quickly respond "*No comment, read problem statement.*"

   You will need to explain carefully what part of the problem statement you think is ambiguous or incorrect and why you think so if you want to convince the judges that a more substantive response is justified.

4. Do not use clarifications to inquire about any test data, including the sample test cases. A clarification request such as "What would the correct output be for this input data?", "In this test case, shall I output X or Y?", or "How do I obtain the answer in the first sample case?" will receive the no comment response. You are expected to figure that out yourself from the problem description.

   You may, however, under very rare circumstances, wish to include test inputs and your anticipated output as part of a careful argument that the problem description is ambiguous or incorrect.

5. Do not use clarifications to ask questions about your local environment (physical or electronic). Since this is a remote contest, you are responsible for all your local environment setup.

6. If, due to unforeseen circumstances, judging for one or more problems begins to lag more than 30 minutes behind submissions, a clarification announcement will be issued to all teams.

7. Notifications that there are new clarification replies are displayed on all Kattis web pages, but not on other pages (e.g., API documentation and scoreboards).

   You may need to manually refresh those pages to see if new notifications have arrived.