JudgeDB Technical Regulations

By submitting solutions to problems on the JudgeDB platform you acknowledge you have read and understood the following.

Platform rules

1. **System administrator.** The administrator is Mateusz Gienieczko, giem@in.tum.de.
2. **Account.** Your user account at JudgeDB must have an accurate first and last name and an up-to-date email address.
3. **Submission limit.** Each task has a limit of 100 submissions. When exceeded by a user, it can be increased by the system administrator on a case-by-case basis.
4. **Judging procedure.** The code of the submission must be contained within a single file and must not exceed 10KiB. Each submission has a unique number which **must** be included when referring to it to the system administrator or course lecturer.

   4.1 The code is compiled depending on the source language according to the system specification (see below). The result is a single executable. Compilation cannot take longer than 30 seconds of wall clock time and the resulting executable must not exceed 5MiB.

   4.2 The executable is executed in a controlled, sandboxed environment on each test case in isolation. Each test may define its own virtual time limit, measured on a virtual CPU (see technical spec). Additionally, there is a real wall clock time limit to prevent runaway execution. The memory limit is the same for each test in a given problem. The test input is passed as the standard input to the program. The standard output of the program is saved. An error can be reported at this stage due to any of the following:

      a) the program terminates with a non-zero exit code, resulting in a `RUNTIME ERROR` report code;
      b) the program dynamically allocates more than the allowed memory limit specified in the problem, resulting in termination and a `MEMORY LIMIT EXCEEDED` report code;
      c) the program exceeds the virtual or real time limit, resulting in termination and a `TIME LIMIT EXCEEDED` report code;
      d) the program may attempt an illegal syscall or attempt to tamper with the sandbox in other ways, resulting in immediate termination and a `RULES VIOLATION` error code.

   4.3 If execution finished successfully the output of the program is compared with the model solution. The points awarded depend on the task specification and may include partial points for partially correct outputs. Correct or partially correct output results in an `OK` report code; incorrect outputs result in a `WRONG ANSWER` error code.

5. **Points awarded.** Tests in a task are organised in one or more test *groups*. Points are awarded per group as the minimum of points awarded over all tests in the group.
   a) As an example imagine two test groups with three tests each, 1a, 1b, 1c; 2a, 2b, 2c; and the task specifies 3 points for group 1 and 7 for group 2. Assume the results for 1a, 1b, 1c, 2a, 2b are `OK`, while 2c had a `WRONG ANSWER`. Therefore the points for each of the tests are 3, 3, 3, 7, 0, respectively, and thus group 1 is awarded with 3 points, group 2 with 0 points, and the overall result is 3 points.
   b) Assume now that instead 2c had a partial correct answer worth 50% of the points. The system then awards 4 points for that test (standard rounding), group 2 now gets 4 points, and the total score is 7.

6. **User report.** While the task is in the active round, the detailed test results are hidden and the user sees only the initial tests of the task (always worth 0 points), the points awarded for the submission, and the overall grading status which is either `OK` if all tests were `OK`, or the non-`OK` report code for the first failed test case. After the results are revealed users can see the full report with each test case listed individually.
8. **System failures.** The system may be slow to judge submissions when under heavy load. However, if the site is not responsive for a long time, or your submission is pending for an exceedingly long time (an hour or more), please report this as an issue directly to the system administrator. In rare cases, the system may report a `SYSTEM ERROR` report code, indicating a non-user error. Such cases should be reported to the system administrator as soon as possible. Always include the submission number when reporting such issues.
9. **Malicious actions.** Attempts to break the system (such as denial-of-service attacks) or break out of the sandbox during execution are not allowed and will result in an account ban.

System specification

1. **Languages and compilers.** The system allows submissions in **Rust**, **C++**, and **C**. Each language has a single compiler used for compiling all submissions in that language. Compilation is sandboxed using a Docker container. All submissions are compiled **on a 32-bit x86 architecture** with optimisations enabled, statically linked, with debug symbols stripped. Specifically:

   - **Rust.** The submission is saved as `a.rs` and then compiled with:

   ```
   rustc -O -Ctarget-feature=+crt-static -Cstrip=symbols -o a.out a.rs
   ```

   No crates outside of the Rust standard are available. The Docker environment used for compilation is the official Rust Docker image `rust:1.86.0-slim` with the platform flag `--platform i386`.

   - **C++.** The submission is saved as `a.cpp` and compiled with:

   ```
   g++ -std=c++23 -static -O2 -s -lm -o a.out a.cpp
   ```

   The g++ version is 14.2.0 and the libc implementation is MUSL. The Docker environment used is defined by the following Dockerfile:

   ```
   FROM alpine:3.21
   RUN apk add build-base musl-dev 'g++=14.2.0-r4'
   ```

   - **C.** The submission is saved as `a.c` and compiled with:

   ```
   gcc -std=gnu17 -static -O2 -s -lm -o a.out a.c
   ```

   The gcc version is 14.2.0 and the libc implementation is MUSL. This environment is identical to the C++ one.
2. **Sandbox.** Executables are run in a sandbox that blocks submissions from performing most syscalls. Programs are **not allowed**, among other things:
   - any kind of networking,
   - to access the file system,
   - to manipulate file descriptors other than the standard input and output,
   - to spawn threads.
3. **Virtual CPU.** Execution is traced and the time is calculated based on a virtual CPU clocked at 2GHz. This is much slower than most modern machines, therefore execution times on local machines are likely to be significantly lower than in the system. However, this technique ensures

that submissions are isolated and the system load does not influence the grading of concurrent submissions.

Data privacy

Your personal data provided as part of registration is stored entirely on TUM machines and processed by TUM employees for the sole purpose of identifying your platform account with your student records. The site is available from the public internet and the ranking for some contests may be viewed by anyone. You can opt out from your name appearing in the ranking at any time by checking the Anonymous box in your contest registration. You can delete your account at any time, which will result in purging of all your user records except as required by TUM for archival purposes.

By submitting code to the platform you agree that it may be processed by outside services for the purposes of plagiarism detection such as Stanford's MOSS. This process is anonymous, i.e. only the code is processed, however if you include any personally identifiable information in your code these may become public - avoid doing so, in particular do not include file headers with your personal data.

by Mateusz Gienieczko, edited 06.10.2025