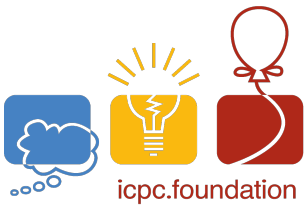

icpc international collegiate programming contest

Technical Notes

Revised: Dec 17, 2024

This document contains important technical information related to the ICPC Asia Regional Contest programming environment. It is important that your team read and understand all the information below.

All Programs

- The languages allowed in the contest are C, C++, Java, Kotlin, and Python 3.
- There is a limit of 256 kibibytes on the length of file submitted for judging.
- Your program must read its input from "standard input".
- Your program should send its output to "standard output". Your program may also send output to "standard error", but only output sent to "standard output" will be considered during judging. (Note that sending too much output to "standard error" might be harmful, in the sense that it can slow your program down.)
- All program source code files and/or test data files which you create must be located in or beneath your "home directory". You may create subdirectories beneath your home directory.
- If your program exits with a non-zero exit code, it will be judged as a **RUN-ERROR**. This can have a lot of different causes like division by zero, incorrectly addressing memory (e.g. by indexing arrays out of bounds), trying to use more memory than the limit, reading or writing to files, etc.
- Programs submitted to the judges will be run inside a "sandbox".
  - The sandbox will allocate memory for your program as specified in each problem.
  - Your entire program, including its runtime environment, must execute within the specified memory limit for the problem. For interpreted languages (Java, Kotlin and Python), the "runtime environment" **includes** the interpreter (that is, the JVM for Java and Kotlin and the Python interpreter for Python).
  - The sandbox memory allocation limit will be the same for every language.
  - The command and command-line arguments used to invoke your program within the sandbox are the same as those given below.
  - Programs running in the sandbox will be "pinned" to a *single* CPU.

C/C++ Programs

- Use the filename extension "`.cpp`" for C++ program files (extensions `.cc`, `.cxx`, and `.c++` can also be used).
- Use the filename extension "`.c`" for C program files.

Java Programs

- **Do not** use package statements (that is, your solution should reside in the "default package").
- Use the filename extension "`.java`" for all Java source files.

Kotlin Programs

- **Do not** use package statements (that is, your solution should reside in the "default package").
- Use the filename extension "`.kt`" for all Kotlin source files.

Python Programs

- In conformance with *World Finals* rules, only Python3 (but not Python2) is allowed.
- Use the filename extension "`.py`" for all Python3 source files.
- Python programs will be "syntax checked" when submitted; programs which fail the syntax check will receive a "**COMPILER-ERROR**" response (for which no penalty applies, just as with C/C++/Java/Kotlin programs which fail to compile). See the sections below for information on how to perform a syntax check yourself in the same way as will be done by the judges.

In the following sections, the notation "`${files}`" means "the list of file names passed to the corresponding script as arguments". For Java and Kotlin submissions, the notation "`${mainclass}`" refers to the name of the main class in your program.

Command-Line Usage

You are free to execute (test) your programs on your machine using any method you choose. However, it is recommended that you compile and execute your programs using the command line scripts described below since this will ensure the closest match to the way in which the judges will compile and execute your programs.

Note that the scripts do **NOT** invoke a sandbox like the judges will do; in particular, the scripts do not enforce memory nor time limits like the judge's sandbox will.

C (GCC 11.3.0) / C++ (G++ 11.3.0)

- To compile a C or C++ program from a command line, type the command

```
compilegcc  progname.c    # for C programs
compileg++  progname.cpp  # for C++ programs
```

  where `progname.c` or `progname.cpp` is the name of your source code file.
- The `compilegcc` command is a script which invokes the GNU GCC compiler with the same options as those used by the judges:

```
gcc -x c -g -O2 -std=gnu11 -fmax-errors=3 -static ${files} -lm
```

- The `compileg++` command is a script which invokes the GNU G++ compiler with the same options as those used by the judges:

```
g++ -x c++ -g -O2 -std=gnu++20 -fmax-errors=3 -static ${files}
```

- To execute a C program after compiling it as above, type the command `runc` ; to execute a C++ program after compiling it as above, type the command `runcpp`.

Java (OpenJDK 17.0.5)

- To compile a Java program from a command line, type the command

  ```
  compilejava Progname.java
  ```

  where `Progname.java` is the name of your source code file. This will compile the source code in the file `Progname.java`, and will produce a class file named `Progname.class`.
- The `compilejava` command is a script which invokes the `javac` compiler with the same options as those used by the judges:

  ```
  javac -encoding UTF-8 -sourcepath . -d . ${files}
  ```
- To execute a Java program after compiling it, type the command

  ```
  runjava Progname
  ```

  where `Progname` is the name of the class containing your `main` method (your source code file name without the filename extension).
- The `runjava` command is a script which invokes the `java` command with same options as those used by the judges:

  ```
  java  -Dfile.encoding=UTF-8 -XX:+UseSerialGC \
    -Xss64m -Xms1920m -Xmx1920m ${mainclass}
  ```

Kotlin (Version 1.7.21)

- To compile a Kotlin program from a command line, type the command

  ```
  compilekotlin progname.kt
  ```

  where `progname.kt` is the name of your Kotlin source code file.
- The `compilekotlin` command is a script which invokes the `kotlinc` compiler with the same arguments as those used by the judges:

  ```
  kotlinc -d . ${files}
  ```
- To execute a Kotlin program from a command line, type the command

  ```
  runkotlin PrognameKt
  ```

  where `PrognameKt` is the name of your Kotlin source code file (note the capitalization and the lack of a period in the `runkotlin` argument.)
- The `runkotlin` command is a script which invokes the Kotlin JVM with same options as those used by the judges:

  ```
  kotlin -Dfile.encoding=UTF-8 -J-XX:+UseSerialGC \
    -J-Xss64m -J-Xms1920m -J-Xmx1920m ${mainclass}
  ```

Python 3 (PyPy 7.3.10 with GCC 9.4.0 providing python 3.9.15)

- To "compile" (syntax-check) a Python3 program from a command line, type the command

```
compilepython3 progname.py
```

where `progname.py` is the name of your Python3 source code file.

- The `compilepython3` command is a script which invokes the `pypy3` Python3 interpreter as follows:

```
pypy3 -m py_compile ${files}
```

which compiles (but does not execute) the specified Python program and displays the result (i.e., whether the compile/syntax-check was successful or not).

- To execute a Python3 program from a command line, type the command

```
runpython3 progname.py
```

where `progname.py` is the name of your Python3 source code file.

- The `runpython3` command is a script which invokes the `pypy3` Python3 interpreter passing to it the specified Python program file.

IDEs and Editors

- The following IDEs (Integrated Development Environments) are available on the contest system: **CLion**, **Code::Blocks**, **Eclipse**, **IntelliJ IDEA**, **PyCharm**, **VS Code**. They can be accessed using the *Applications* menu.
- The following editors are available on the contest system: **Vim**, **Gvim**, **Emacs**, **Text Editor (GEdit)**, **Geany**, **Kate**. They can be accessed using the *Applications* menu.

Programming Language Documentation

- Documentation for each available programming language can be found on your machine under the *Applications* > *Programming* > *Documents* menu.

Submission

- Programs are submitted to the judges using the DOMjudge contest control system.
- To access DOMjudge, use the *Applications* > *Contest* > *DOMjudge* menu item.
- See the separate *DOMjudge Team Guide* for details on using DOMjudge.
- Any behavior that is deemed an activity detrimental to the contest may result in disqualification, determined by the staff.

Printing

- There will be runners who will deliver printed output to your team workstation (teams will not have direct access to the printers).
- To print a file, upload it and submit a job in DOMjudge **`Print`** page. Only source code (plain text) can be printed. Please **DO NOT** submit non-text files (e.g. ZIP, PDF, etc.).
- Printing files from within IDEs and other applications via system dialog is **NOT supported**.

- Print jobs are limited to a few pages long; printing excessively long output will be deemed an activity detrimental to the contest and subject to disqualification.

Sample data and Problem Statements

- Sample data for each problem will be provided in DOMjudge `Problem Set` page.
- Time and memory limit for each problem can be viewed in DOMjudge `Problem Set` page.

Files and Data Storage

- Any files that you create must be stored underneath your home directory (this does not apply to files automatically created by system tools such as editors). Your machine is scheduled to be reset prior to the start of the actual contest; any files you create or system configuration changes you make prior to that will be removed as part of this process.