The 2023 ICPC World Finals | Luxor

icpc.foundation icpc International Collegiate Programming Contest hosted by AASTMT

TECHNICAL NOTES

This document contains important technical information related to the ICPC World Finals environment. It is important that your team read and understand all the information below.

All Programs:

- 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”. Your home directory will be named “/home/team*X*”, where “*X*” is your team number. 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 Time Error.
- Programs submitted to the judges will be run inside a “sandbox”.
    - ➢ The sandbox will allow allocation of up to 2GB of memory for your program. Your entire program, including its runtime environment, must execute within this memory limit. 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 size will be the same for every language and problem.
    - ➢ The command and command-line arguments used to invoke your program within the sandbox will be the same as those given below.
    - ➢ Programs running in the sandbox will be “pinned” to a *single* CPU.

All 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.

All Java Programs:

- We suggest that you do not use package statements (that is, we suggest that your solution reside in the “default package”.) Use the filename extension “**.java**” for all Java source files.

All Kotlin Programs:

- We suggest that you do not use package statements (that is, we suggest that your solution reside in the "default package"). Use the filename extension "**.kt**" for all Kotlin source files.

All Python Programs:

- 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 "Compilation 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.

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.

In particular, the **runxxx** scripts below will pin your program's execution to a single CPU just as the judges will do. (Note however 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.)

Command-line Usage for C/C++:

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

```
compilegcc    progname.c        (for C programs)  or

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:

```
-x c -g -O2 -std=gnu11 -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:

```
-x c++ -g -O2 -std=gnu++20 -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`**.

Command-line Usage for Java:

- 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:

  ```
  -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 the following options (which are identical to what the judges will use):

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

Command-line Usage for Python3:

- 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.

- Note that the above commands are precisely what the judges will use to compile and execute Python3 submissions.

Command-line Usage for Kotlin:

- 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:

  ```
  -d . ${files}
  ```

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

  ```
  runkotlin  PrognameKt
  ```

  where **`progname.kt`** 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 the following options (which are identical to what the judges will use):

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

Submissions

- 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.

Language Documentation

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

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, we recommend using the following command:

  ```
  printfile filename
  ```

  where **`filename`** is the name of the file you want printed. Note: printing using other means, for example from within an IDE, *may* work – but this is not guaranteed.

- 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

- Sample data for each problem will be provided in the *samps* directory under the *Desktop* directory beneath your home directory. If there are testing tools for interactive problems (see the separate *Addendum to Judging Notes: Interactive Problems*), then the testing tools will also appear in the corresponding *samps* directory.

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 will be reset prior to the start of the World Finals; any files you create or system configuration changes you make prior to that will be removed.
- In the event of the need to pause the contest for unforeseen reasons, team desktops will be screen-locked. Data already saved on disk should not be affected by this process and should still be available once the contest is resumed and the desktops are unlocked. However, any virtual consoles will be killed during the pause. Teams should save files to disk frequently, and should not use virtual consoles for any data they wish to retain.

Obtaining Files After The Contest

- The contents of your home directory (including subdirectories) will be uploaded to the ICPC web site after the contest and can then be accessed by your coach.