



TECHNICAL NOTES

This document contains important information related to the ACM-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.
- 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 allocate 2GB of memory. 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 for every contest problem.
 - The command and command-line arguments used to invoke your program within the sandbox are the same as those given below. For Java and Kotlin, the command-line arguments explicitly set both the heap size and the stack size. For C, C++, and Python, the heap and stack sizes are limited only by the total amount of memory available.

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 Python Programs:

- You may use either Python 2 or Python 3, but note that the contest environment treats these as two “different languages”. Be sure to submit your program specifying the correct “language”. Use the filename extension “.py” for all Python 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.

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.

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:

```
-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:

```
-g -O2 -std=gnu++14 -static ${files}
```

- To execute a C/C++ program after compiling it as above, type the command

```
./a.out
```

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 Programe
```

where **Programe** 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 Python 2:

- To “compile” (syntax-check) a Python 2 program from a command line, type the command

```
compilepython2 programe.py
```

where **programe.py** is the name of your Python 2 source code file. The **compilepython2** command is a script which invokes the **CPython** Python 2 interpreter as follows:

```
python2 -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 Python 2 program from a command line, type the command

```
runpython2 programe.py
```

where **programe.py** is the name of your Python 2 source code file. The **runpython2** command is a script which invokes the **pypy** Python 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 Python 2 submissions.

Command-line Usage for Python 3:

- To “compile” (syntax-check) a Python 3 program from a command line, type the command

```
compilepython3 programe.py
```

where **programe.py** is the name of your Python 3 source code file. The **compilepython3** command is a script which invokes the **CPython** Python 3 interpreter as follows:

```
python3 -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 Python 3 program from a command line, type the command

```
runpython3 progname.py
```

where `progname.py` is the name of your Python 3 source code file. The `runpython3` command is a script which invokes the `CPython` Python 3 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 Python 3 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 steps described in the *DOMJudge Team Guide*; see that separate document for details.

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). *The runners will only deliver output printed using the `printfile` command* (see below). Note in particular that this means printing using commands from within IDEs is not supported.

- To use `printfile`, go to a command prompt (open a shell terminal window) and type

```
printfile filename
```

where `filename` is the name of the file you want printed. 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 home 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 should not be affected 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 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.