TECHNICAL NOTES

This document contains important information related to the ICPC North America Regionals 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 256K bytes on the total length of files 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”. 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 r*un-rime error*.
- Your program should not use any “architecture-specific” operations (such as invoking *pragma* directives to target a specific architecture or optimization level). The architecture of the machine on which the judges will run your program will be similar to, *but not necessarily identical to*, your team’s machine.
- Programs submitted to the judges will be run inside a “*sandbox*”.
  - The sandbox will allow allocation of memory for your program up to the specified limit in each problem statement (by default 2GiB).
  - 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:

- We strongly suggest that you **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:

- We strongly suggest that you **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:

- Only Python 3 (but not Python 2) is supported. Use the filename extension "**.py**" for all Python3 source files.

- Python3 programs will be "syntax checked" when submitted; programs which fail the syntax check will receive a *compiler error* judgement 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:

Note: 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.

- You are free to execute (test) your programs on your machine using any method you choose. However*, it is strongly recommended that you compile and execute your programs using the command line scripts described below*, which are precisely what the judges will use to compile and execute submissions. Testing your program with these scripts will ensure the closest match to the way in which the judges will compile and execute your programs.

- In particular, note that 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++23 -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`**.

- The contest image has *two* different versions of the GNU C/C++ compilers installed. The PC^2 contest control system judging system uses **`gcc-14`** for C and **`g++-14`** for C++ which use version 14.2.0 of the GNU compilers. The **`compilegcc`** and **`compileg++`** commands use version 14.2.0 of the GNU compilers. It should be noted that the **`gcc`**/**`g++`** commands will use version 13.2.0 of the GNU compilers.

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 same options as those used by the Judges:

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

Command-line Usage for Python 3:

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

  `compilepython3  progname.py`

  where **`progname.py`** is the name of your Python 3 source code file. The **`compilepython3`** command is a script which invokes the **`PyPy3`** Python 3 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 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 **`pypy3`** 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}
  ```

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(GUI)**, **Emacs(Terminal)**, **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⇨Documentation⇨Development Language Tools* menu.

Submissions

- Programs are submitted to the judges using the *PC²* contest control system. Submissions may be made using the PC² web interface or **`pc2submit`** from the command line. To access the *PC²* web interface, use the *Applications⇨Contest⇨CCS* menu item. See the separate PC² *Team Guide* for details on using *PC².*

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 from a shell terminal window:

  ```
  printfile filename
  ```

  where **`filename`** is the name of the file you want printed. Note: printing files using other means, such as from within IDEs and other applications, *may* work, but 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 and Problem Statements

- PDF Problem Statements and Sample data for each problem will be provided using the *Applications⇨Documentation⇨Contest Problem Set* menu item or the *Problem Set* shortcut on the web browser address bar. If there are testing tools for interactive problems (see the separate *Judging Notes*), then the testing tools will also appear on the corresponding *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 will 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.

- In the event of the need to pause the contest for unforeseen reasons, new team submissions and access to the contest server will be disabled. Data already saved on disk should not be affected by this process and should still be available once the contest is resumed. It is always a good idea for teams to save files to disk frequently.

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.