



TECHNICAL NOTES

Subject to Change
Updated: November 1, 2024

This document contains important information related to the environment for the Greater New York Regional Contest. It is important that your team read and understand all the information below.

The contest environment for the Greater New York Regional contest boots off of a USB 3.1 thumb drive at least 32GB in size. This *Ubuntu 24.04 LTS* based environment is essentially the same one that will be used at the *ICPC World Finals* in 2025. Some monitoring tools are in-place to help ensure a secure contest. This is a new image, and the first time it is being used at a regional contest.

All Programs:

- The languages allowed in the contest are C, C++, Java, Kotlin, and Python 3.
- There is a limit of 65,535 bytes on the 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.
- 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 normally be named “/home/team” – but this may vary between sites so check with your site coordinator. 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.
- Each problem will specify a maximum CPU time limit your submission can use for a single test case. If the CPU time limit is exceeded, it will be judged as Time Limit Exceeded.
- Each problem will specify a maximum RAM (memory) limit your submission can use for a single test case. If the RAM limit is exceeded, it will be judged as Memory Limit Exceeded.

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. Be sure to capitalize (at least) the first letter of the filename of your Kotlin source file.

Python Programs:

- In conformance with World Finals rules, only Python 3 (but not Python 2) is supported (see information below about which version of Python 3 is supported). 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” judgment response (for which no penalty currently 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.

Note: in the following sections, the notation “**#{files}**” means “the list of file names passed to the corresponding script as arguments”. We strongly encourage you to compile your programs using the commands below, since these commands will use the same compiler options as the judging systems. Please note that on the judging systems, your submission will be “*pinned*” to a single CPU (using the `taskset` command) - keep this in mind if you receive Time Limit Exceeded errors.

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/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 Progame.java
```

where `Progame.java` is the name of your source code file. This will compile the source code in the file `Progame.java`, and will produce a class file named `Progame.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 Progame
```

where `Progame` 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 Progame.kt
```

where `Progame.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 ProgameKt
```

where `progame.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 from the **Applications->Development** menu: **CLion, Code::Blocks, Eclipse, IntelliJ IDEA, PyCharm, VS Code.**
- The following editors are available on the contest system from the **Applications->Development** menu: **Vim, Gvim, Emacs(GUI), Emacs(Terminal), GEdit, Geany, Kate.**

Documentation

- Documentation for each available programming language can be found on your machine using the “**Documentation**” bookmark provided on the address bar in the Firefox web browser..

Submissions

- Programs are submitted to the Judges using the PC^2 contest control system *Web Team Interface (WTI)*. The *WTI* can be accessed using the URL provided in the credentials Email you received.

Scoreboard

- The current contest scoreboard (standings) can be viewed using the *Scoreboard* tab item on the PC^2 *WTI*. Note that the scoreboard will be *frozen* with **one hour** remaining in the contest **or** if at least one team has solved *one less* than the total number of problems.



Printing

- Teams may print source code listings at their Microsite, if printing is available there. We will not be able to provide support in setting this up.
- Specific additional instructions for printing may vary once on-site; ask your site coordinator for additional information regarding printing at your site.

Network Connection

- You must be able to connect to the internet to make submissions using the Web Team Interface (*WTI*). The *WTI* site is the **ONLY** site you are permitted to access during the contest. The contest image is configured to allow a hardwired ethernet connection (just plug in an RJ45 connector), and it will connect to the network using DHCP.
- You may also configure WIFI on the contest image. There are settings available on the desktop to do this easily. Look for the WIFI icon just to the left of the date and time on the upper right of your display.

Sample data and Problem Statements

- Sample data and problem statements for each problem will be accessible in machine-readable form via a link provided in your credential Email. You will be able to read or download individual items or a single *.zip* file containing the sample data and problem statements for all problems in the contest. You can *unzip* that file under your home directory where an individual folder for each problem will be created.

Files and Data Storage

- Any files that you create must be stored underneath the home directory of the **team** account (this does not apply to files automatically created by system tools such as editors).

Language and Tool Versions

- The contest image, as well as the judging system uses the following compiler and tool versions. If you are using a version of a compiler that is different from these, we can not help you diagnose issues. If you are using a system other than the designated contest image, we can not help diagnose issues. Compile or run-time errors can often be traced back to compiler version differences.

Compiler/Tool	Version
g++ (C++)	13.2.0
gcc (C)	13.2.0
java	21.0.4
kotlin	1.9.24
pypy3	7.3.15 with Python 3.9.18
Eclipse	2024-09 (4.33.0)
Jetbrains CLion	2024.2.2
Jetbrains IntelliJ	2024.2.2
Jetbrains PyCharm	2024.2.3

Special Thanks

Special thanks to Dr. John Clevenger (*ICPC World Finals* Technical Director) for writing the original version of this document. Any mistakes in here are mine, not his. – John Buck