Judging Details

Judge System

Your programs will be judged on the system once they're submitted.

- Your program must read input data from the standard input, and write its output to the standard output.
- Other outputs, e.g. writing to the standard error, will not be used for judging.
- You will never have to write to (open) a file, and are not allowed to do so.

Your programs will be run inside a *sandboxed environment*, i.e. with protections to prevent the system from being damaged. Specifically:

- Memory usage is limited to 2 GB in the environment. Note it is the total amount, not the amount you can use exclusively in your programs.
- The stack size is set unlimited (in C/C++), only capped by the total memory limit.
- Multi-processing or multi-threading is discouraged and unlikely beneficial, though not prohibited. Remember your programs will run on a single processor core. The total number of processes is limited to 64, including ones the system may create outside your programs.
- It is *never* recommended to run external commands. It is technically possible but probably does not work as you expect.

If you have no idea about what these mean — no worries. Just remember your programs should use the standard input and output, not files.

There are a couple more restrictions that apply:

- The total amount of source code must not exceed 256 KB in each submission.
- Your program must compile within 30 seconds.

See the DOMjudge team manual for more details about these restrictions.

Note about Platform

The judge system is running on Google Compute Engine, C2 machine type (`c2-standard-4`). For more information about Google Compute Engine, please visit the official website[*1].

1. https://cloud.google.com/compute/docs/cpu-platforms

Compilers & Options

The judge system uses the following compilers and execution environments (e.g., interpreters) with the following options. `"$@"` is substituted with your source file(s); `"$DEST"` is the name of the binary (which is `./a.out` by default) and is chosen arbitrarily by the system.

The **Run** commands indicated in the following table are for non-interactive problems. For interactive problems, standard input and output are connected to a judge program. See the "Note on Interactive Problems" section below for the details.

C	
Version	gcc (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0
Compile	`gcc -x c -g -O2 -std=gnu11 -static -o "$DEST" "$@" -lm`
Run	`"$DEST" < infile > outfile`
C++	
Version	g++ (Ubuntu 11.4.0-1ubuntu1~22.04) 11.4.0
Compile	`g++ -x c++ -g -O2 -std=gnu++20 -static -o "$DEST" "$@"`
Run	`"$DEST" < infile > outfile`
Java	
Version	OpenJDK 17.0.8.1 (build 17.0.8.1+1-Ubuntu-0ubuntu122.04)
Compile	`javac -encoding UTF-8 -sourcepath . -d . "$@"`
Run	`java -Dfile.encoding=UTF-8 -XX:+UseSerialGC -Xss64m -Xms1920m -Xmx1920m MainClass < infile > outfile`
Python 3 (PyPy)	
Version	Python 3.8.13 (7.3.9+dfsg-1, Apr 01 2022, 21:41:47) [PyPy 7.3.9 with GCC 11.2.0]
Compile	`pypy3 -m py_compile "$@"`
Run	`pypy3 "$@" < infile > outfile`
Kotlin	
Version	1.7.21 (JRE 17.0.8.1+1-Ubuntu-0ubuntu122.04)
Compile	`kotlinc -d . "$@"`
Run	`kotlin -Dfile.encoding=UTF-8 -J-XX:+UseSerialGC -J-Xss64m -J-Xms1920m -J-Xmx1920m MainClass < infile > outfile`

In Java and Kotlin, DOMjudge will detect the main class automatically; you do not have to name it `Main`. See the DOMjudge team manual for details.

In Python, **Compile** commands only verify the syntax. `*.pyc` files will not be used in the real run.

The compilers and the execution environments are also available on your workstation as the following commands:

- **C** — `compilegcc` / `runc`
- **C++** — `compileg++` / `runcpp`
- **Java** — `compilejava` / `runjava`
- **Python 3** — `compilepython3` / `runpython3`
- **Kotlin** — `compilekotlin` / `runkotlin`

Submission Results

The judges may have prepared multiple test cases for each problem. On each submission, DOMjudge decides one result for each test case. DOMjudge does *not* report results for each test case, but it reports one result for a submission, based on the following rules.

Results for test cases

For each test case, DOMjudge decides one of the following results:

- **CORRECT** - Your program ran successfully and passed the test case.
- **TIMELIMIT** — Your program did not finish within the time limit.
- **RUN-ERROR** — Your program crashed or exited with a non-zero exit status (e.g. because of missing `return 0;` in C/C++).
- **OUTPUT-LIMIT** — Your program produced excessive output (> 8 MB).
- **WRONG-ANSWER** — Your program neither crashed nor exceeded the time limit, but produced incorrect output.
- **NO-OUTPUT** — Your program did not produce any output.

See the DOMjudge team manual for more details about these results.

Results for submissions

For each submission, DOMjudge reports one of the following results:

Accepted

- **CORRECT** — Your program resulted in **CORRECT** for all test cases.

Rejected with 20-minutes penalty

- **TIMELIMIT**, **RUN-ERROR**, **OUTPUT-LIMIT**, **WRONG-ANSWER**, **NO-OUTPUT** — If your program resulted in **TIMELIMIT**, **RUN-ERROR**, **OUTPUT-LIMIT**, **WRONG-ANSWER** or **NO-OUTPUT** for any test case, then that result is returned immediately.

Rejected with no penalty

The following results imply your program did not even start. You do not receive any penalty for these results.

- **COMPILE-ERROR** — Your program did not compile in the judging environment. You can consult the error message(s) on the submission details page.
- **TOO-LATE** — Your program was submitted after the contest was over. [*2]

Note on Interactive Problems

You may meet “interactive problems” in the contest. They are the same as other problems in a way that your program will read from standard input and print results to standard output. The difference is, the standard input and output are connected to a special program (judge program), with which you have to communicate back and forth. Unlike other problems where the input text is fixed for each test case, the input varies based on your previous outputs.

In most programming environments, program output is buffered to speed up I/O operations. With interactive problems, it is crucial to make sure the output is actually sent from your program and not simply stored in internal buffers. This typically means flushing the output buffers after each write.

- In C/C++ with `stdio.h` (or `cstdio`), you can use `fflush(stdout)`. Writing `\n` does not mean it will get flushed.
- In C++ with `iostream`, an output stream is flushed automatically each time you write the `std::endl` manipulator. When using other means or if you want to be sure, call `std::cout.flush()`.
- In Java and Kotlin, the `System.out` stream has so-called “auto-flush” functionality and its buffer is therefore flushed automatically with each newline character. When using other streams or if you want to be sure, invoke the `flush()` method of the stream.
- In Python, you can use `sys.stdout.flush()`.

The time limit for an interactive problem is how much time your submission may spend; the time spent by the judge program is *not* counted towards this. Note that if your program attempts to read more input than can be provided currently (e.g., because you forgot to flush your previous output, or because of some other reason), then the program will stall indefinitely and your submission will get **TIMELIMIT**.

Note on Languages

The judges have solved all problems in languages from at least two of the three distinct language groups (Java/Kotlin, C/C++, and Python).

Note to Python Users

Only syntax errors will be reported as **COMPILE-ERROR**. Other types of errors, such as `NameError` or `ModuleNotFoundError`, will result in **RUN-ERROR** and incur a 20-minute penalty.

It is fine, though not needed, to start your scripts with an interpreter directive (line starting with `#!`, also known as shebang). [*3]

The full list of modules available in the judge system can be found in the following section.

---

2. Note that this does not mean your programs need to be judged before the end of the contest. Your programs will be judged as long as submitted (“queued”) within the contest time.
3. Some past versions of DOMjudge refused scripts that contain a shebang.

Available Python Modules

```
__decimal           _rawffi             ensurepip           pypy_tools
__exceptions__      _resource_build     enum                pypyjit
__future__          _resource_cffi      errno               pyrepl
__pypy__            _scproxy            faulthandler        queue
_abc                _sha1               fcntl               quopri
_ast                _sha256             filecmp             random
_audioop_build      _sha3               fileinput           re
_audioop_cffi       _sha512             fnmatch             readline
_blake2             _signal             formatter           reprlib
_bootlocale         _sitebuiltins       fractions           resource
_bz2                _socket             ftplib              rlcompleter
_cffi_backend       _sqlite3            functools           runpy
_cffi_ssl           _sqlite3_build      future_builtins     sched
_codecs             _sqlite3_cffi       gc                  secrets
_codecs_cn          _sre                genericpath         select
_codecs_hk          _ssl                getopt              selectors
_codecs_iso2022     _ssl_build          getpass             setuptools
_codecs_jp          _string             gettext             shelve
_codecs_kr          _strptime           glob                shlex
_codecs_tw          _struct             greenlet            shutil
_collections        _structseq          grp                 signal
_collections_abc    _sysconfigdata      gzip                site
_compat_pickle      _syslog_build       hashlib             smtpd
_compression        _syslog_cffi        heapq               smtplib
_contextvars        _testcapi           hmac                sndhdr
_continuation       _testing            html                socket
_cppyy              _testmultiphase     http                socketserver
_crypt              _thread             identity_dict       sqlite3
_csv                _threading_local    idlelib             sre_compile
_ctypes             _vmprof             imaplib             sre_constants
_ctypes_test        _warnings           imghdr              sre_parse
_curses             _weakref            imp                 ssl
_curses_build       _weakrefset         importlib           stackless
_curses_cffi        _winapi             inspect             stat
_curses_panel       abc                 io                  statistics
_dbm                aifc                ipaddress           string
_decimal_build      antigravity         itertools           stringprep
_distutils_hack     argparse            json                struct
_distutils_system_mod array               keyword             subprocess
_dummy_thread       ast                 lib2to3             sunau
_ffi                asynchat            linecache           symbol
_functools          asyncio             locale              symtable
_gdbm               asyncore            logging             sys
_gdbm_build         atexit              lzma                sysconfig
_gdbm_cffi          audioop             macpath             syslog
_hashlib            base64              macurl2path         tabnanny
_hpy_universal      bdb                 mailbox             tarfile
_immutables_map     binascii            mailcap             telnetlib
_imp                binhex              marshal             tempfile
_io                 bisect              math                termios
_jitlog             builtins            mimetypes           test
_locale             bz2                 mmap                textwrap
_lsprof             cProfile            modulefinder        this
_lzma               calendar            msilib              threading
_lzma_build         cffi                msvcrt              time
_lzma_cffi          cgi                 multiprocessing     timeit
_markupbase         cgitb               netrc               tkinter
_marshal            chunk               nntplib             token
_md5                cmath               ntpath              tokenize
_minimal_curses     cmd                 nturl2path          tputil
```

```
_multibytecodec     code                numbers             trace
_multiprocessing    codecs              opcode              traceback
_opcode             codeop              operator            tracemalloc
_operator           collections         optparse            tty
_osx_support        colorsys            os                  turtle
_overlapped         compileall          parser              turtledemo
_pickle_support     concurrent          pathlib             types
_posixshmem         configparser        pdb                 typing
_posixshmem_build   contextlib          pickle              unicodedata
_posixshmem_cffi    contextvars         pickletools         unittest
_posixsubprocess    copy                pip                 urllib
_pwdgrp_build       copyreg             pipes               uu
_pwdgrp_cffi        cpyext              pkg_resources       uuid
_py_abc             crypt               pkgutil             venv
_pydecimal          csv                 platform            warnings
_pyio               ctypes              plistlib            wave
_pypy_interact      ctypes_support      poplib              weakref
_pypy_irc_topic     curses              posix               webbrowser
_pypy_openssl       dataclasses         posixpath           wsgiref
_pypy_testcapi      datetime            pprint              xdrlib
_pypy_util_build    dbm                 profile             xml
_pypy_util_cffi     decimal             pstats              xmlrpc
_pypy_util_cffi_inner difflib             pty                 zipapp
_pypy_wait          dis                 pwd                 zipfile
_pypy_winbase_build distutils           py_compile          zipimport
_pypy_winbase_cffi  doctest             pyclbr              zlib
_pypy_winbase_cffi64 dummy_threading     pydoc
_pypyjson           email               pydoc_data
_random             encodings           pyexpat
```