Judging Notes

1. Judge System

When you submit source code, the judge system decides a verdict by compiling and running it against undisclosed test inputs.

1.1. Input and Output

- Your program must read all input from **standard input** (stdin, e.g., `std::cin` in C++, `System.in` in Java, `sys.stdin` in Python).
- Your program must write all its output to **standard output** (stdout, e.g., `std::cout` in C++, `System.out` in Java, `sys.stdout` in Python).
- Output written to **standard error** (stderr, e.g., `std::cerr` in C++, `System.err` in Java, `sys.stderr` in Python) is ignored when judging.
- Your program is not allowed to open, create, or write to files. These file operations will result in a **RUN-ERROR**.
- Your program must exit with status code 0. Exiting with a non-zero status code will result in a **RUN-ERROR**, even if the output is correct.

1.2. Execution Environment (Sandbox)

Your program will be run inside a *sandboxed environment*. This environment isolates your program to prevent system damage and to ensure equal conditions for all contestants. This means several restrictions apply:

- **Memory Usage:** Total memory usage is limited to **2 GB**. This limit applies to the whole processes, including your program's code, static variables, stack, heap, and any virtual machine overhead (like for Java or Python). Exceeding this limit will typically result in a **RUN-ERROR**.
- **Stack Size:** In C/C++, the stack size is not explicitly limited, but is limited by the total memory limit. In Java/Kotlin, the stack size is limited by the `-Xss` command line parameter. In Python, the recursion depth is limited by the runtime's default recursion limit (1000). Your program can raise this limit by calling `sys.setrecursionlimit()`, but the stack size is still ultimately limited by the total memory limit.
- **Processes/Threads:**
  - Your program will be run on a **single processor core**. Using multiple threads or processes is allowed but generally discouraged as it is unlikely to improve performance.
  - The total number of processes allowed for a run is limited to **64**. This includes the main process of your program and any child processes, as well as processes the system might use internally. Exceeding this limit will result in a **RUN-ERROR**.
- **External Commands:** Running external commands (e.g., via `system()` in C/C++) is restricted and is unlikely to work as expected.

- **File System Access:** As mentioned, opening, creating, or writing to files is not allowed.

1.3. Submission and Compilation

Your submission must meet these requirements:

- **Submission Files:** A single submission can contain one or more files. Each file name must start with an ASCII alphanumeric character (a "half-width" `A`-`Z`, `a`-`z`, or `0`-`9`) and may contain only ASCII alphanumeric characters and `+.\_-`.
- **Source Code Size:** The total size of all source code files in a single submission must not exceed **256 KB**.
- **Source Code Encoding:** All source code files must be encoded in UTF-8. Submitting files with other encodings (e.g., Shift_JIS) may result in a **COMPILER-ERROR** or unexpected behavior.
- **Compilation Time:** Your submission must compile successfully within **30 seconds**. A compilation taking longer than this will be aborted, resulting in a **COMPILER-ERROR**.
- **Compilation Memory Usage:** Total memory usage during compilation is limited to **2 GB**. A compilation exceeding this will be aborted, resulting in a **COMPILER-ERROR**.

1.4. About the Platform

The judge system runs your program on Google Compute Engine, C4 machine type (`c4-highcpu-2`, with `turboMode` set to `ALL_CORE_MAX`). For more information about Google Compute Engine, please visit the official website[*1].

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

2. Compilers & Options

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

The **Run** commands in the following table are for non-interactive problems. For interactive problems, see the "Note on Interactive Problems" section for details.

C	
Version	gcc 13
Compile	`gcc -x c -g -O2 -std=gnu11 -static -o "$DEST" "$@" -lm`
Run	`"$DEST" < infile > outfile`
C++	
Version	g++ 13
Compile	`g++ -x c++ -g -O2 -std=gnu++20 -static -o "$DEST" "$@"`
Run	`"$DEST" < infile > outfile`
Java	
Version	OpenJDK 21
Compile	`javac -encoding UTF-8 -sourcepath . -d . "$@"`
Run	`java -Dfile.encoding=UTF-8 -XX:+UseSerialGC -Xss64m -Xms1920m -Xmx1920m MainClass < infile > outfile`
Python 3 (PyPy)	
Version	Python 3.9.18 (PyPy 7.3.15)
Compile	`pypy3 -m py_compile "$@"`
Run	`pypy3 "$@" < infile > outfile`
Kotlin	
Version	1.9.24
Compile	`kotlinc -d . "$@"`
Run	`kotlin -Dfile.encoding=UTF-8 -J-XX:+UseSerialGC -J-Xss64m -J-Xms1920m -J-Xmx1920m MainClass < infile > outfile`

For Java and Kotlin submissions, the system automatically detects the main class. You do not need to name your main class `Main`.

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

The compilers and the execution environments are also available on your machine, as the following commands (see Appendix A of the "How to Use the System" document for usage):

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

3. Verdicts of Submissions

The judge system runs your program against multiple inputs prepared by the judges.

The system first decides the verdict for each input against your submission. Then, based on those verdicts, the system decides the submission's overall verdict. You will only see this overall verdict, not the verdict for each input.

3.1. Possible Verdicts on Each Input

For each input, the judge system decides one of the following verdicts:

- **CORRECT:** The judge system determined that your program completed its run within the time limit and wrote output that was validated as correct.
  - The validation method is prepared by the judges. This may involve comparing the output to a specific answer or using a custom program for problems where the correct output is not unique.
  - Minor differences in whitespace are ignored in common validation. For example, an extra whitespace at the end of a line or an extra blank line at the end of the output can be ignored.
  - Differences between uppercase and lowercase letters are ignored. For example, `Yes` and `yes` are treated the same.
  - For problems where the correct output may not be unique (e.g., with floating-point answers), the specific validation method will be documented in the problem statement.
- **TIMELIMIT (Time Limit Exceeded):** Your program did not complete the run within the per-problem time limit and was aborted.
- **RUN-ERROR (Runtime Error):** Your program terminated abnormally during the run. Common causes include:
  - Attempting invalid operations (e.g., division by zero).
  - Accessing invalid memory (e.g., array index out of bounds, null pointer dereferencing).
  - Exceeding the memory limit (2 GB).
  - Aborting by `assert()`.
  - Exiting with a non-zero status code (e.g., forgetting `return 0;` at the end of `main` in C). The exit status code is checked *before* the output is checked.
  - Crashing for any other reason.
- **WRONG-ANSWER:** Your program completed the run within the time limit without crashing, but wrote output that was not validated as correct.
- **OUTPUT-LIMIT (Output Limit Exceeded):** Your program wrote output exceeding **8 MB**. This limit does not apply to output written to standard error.
- **NO-OUTPUT:** Your program completed the run successfully within the time limit without crashing but wrote nothing to standard output.

3.2. Overall Verdict of a Submission

The judge system decides one overall verdict of a submission as follows:

3.2.1. Accepted

- **CORRECT:** Your program resulted in **CORRECT** for all inputs.

3.2.2. Rejected (with 20 Minute Penalty)

If the system decides any of the five verdicts below for an input, the system decides that verdict as the overall verdict of the submission, and it will incur a **20-minute time penalty** per submission if the problem is eventually solved.

- **TIMELIMIT**
- **RUN-ERROR**
- **WRONG-ANSWER**
- **OUTPUT-LIMIT**
- **NO-OUTPUT**

If multiple inputs result in different verdicts including ones from this list (e.g., one **TIMELIMIT** and one **WRONG-ANSWER**), the system decides one of the verdicts from the list as the overall verdict of the submission. In this case, there is no guarantee which one will be selected.

3.2.3. Rejected (with No Penalty)

Two verdicts below mean that the system did not run your program against the inputs. They **do not incur any time penalty**.

- **COMPILER-ERROR:** The judge system failed to compile your source code. You can see the compiler's error messages on the submission details page.
- **TOO-LATE:** The judge system received your submission after the contest ends. Note that submissions received before the contest ends are judged normally, even after the contest ends, and they do not receive a **TOO-LATE** verdict.

4. Note on Interactive Problems

You may encounter interactive problems in the contest. Like standard problems, your program will read from standard input and write to standard output. The key difference is that the standard input and output are connected to a special judge program, requiring your program to communicate back and forth with it. In contrast to standard problems where the input is fixed, the input in an interactive problem may change based on the outputs that your program previously wrote.

In most programming environments, program output is buffered to enhance I/O performance. For interactive problems, your output must be sent from your program immediately, rather than being held in these internal buffers. This typically means you must explicitly **flush** the output buffer after each piece of information you write to the standard output.

- C/C++ (`stdio.h` / `cstdio`): Use `fflush(stdout);`. Writing `\n` does not guarantee a flush.
- C++ (`iostream`): Writing `std::endl` automatically flushes the output stream (e.g., `std::cout`) upon writing a newline. If you write a newline using `\n` instead (which does not guarantee a flush), explicitly call `std::cout.flush();`. Note that `std::cin.tie(nullptr)` stops `std::cout` from being flushed automatically before each read from `std::cin`.
- Java / Kotlin: `System.out` is configured for "auto-flush" by default, meaning `println()` flushes the buffer upon writing a newline. When using other streams, invoke the `flush()` method on your output stream or writer.
- Python: Use `sys.stdout.flush()` after your `print()` statements. Alternatively, you can use the `flush=True` argument: `print(..., flush=True)`.

The time limit for an interactive problem applies to the time used by your program; 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 by the judge program at that time (e.g., because you forgot to flush your previous output, or because of some other reason), then your program will stall indefinitely and your submission will result in a **TIMELIMIT**.

5. Note on Languages

For every problem, the judges have verified solutions in languages from at least two of the three distinct language groups (Java/Kotlin, C/C++, and Python).

6. Note to Python Users

Only syntax errors will be reported as a **COMPILER-ERROR**. Other errors, such as `NameError` or `ModuleNotFoundError`, will result in a **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 a shebang).

6.1. Available Python Modules

```
__decimal           _rawffi             email               pyexpat
__exceptions__      _resource_build     encodings           pypy_tools
__future__          _resource_cffi      ensurepip           pypyjit
__pypy__            _scproxy            enum                pyrepl
_abc                _sha1               errno               queue
_aix_support        _sha256             faulthandler        quopri
_ast                _sha3               fcntl               random
_audioop_build      _sha512             filecmp             re
_audioop_cffi       _signal             fileinput           readline
_blake2             _sitebuiltins       fnmatch             reprlib
_bootlocale         _socket             formatter           resource
_bootsubprocess     _sqlite3            fractions           rlcompleter
_bz2                _sqlite3_build      ftplib              runpy
_cffi_backend       _sqlite3_cffi       functools           sched
_cffi_ssl           _sre                future_builtins     secrets
_codecs             _ssl                gc                  select
_codecs_cn          _ssl_build          genericpath         selectors
_codecs_hk          _string             getopt              shelve
_codecs_iso2022     _strptime           getpass             shlex
_codecs_jp          _struct             gettext             shutil
_codecs_kr          _structseq          glob                signal
_codecs_tw          _sysconfigdata      graphlib            site
_collections        _sysconfigdata__linux_x86_64-linux-gnu greenlet            smtpd
_collections_abc    _syslog_build       grp                 smtplib
_compat_pickle      _syslog_cffi        gzip                sndhdr
_compression        _testcapi           hashlib             socket
_contextvars        _testing            heapq               socketserver
_continuation       _testmultiphase     hmac                sqlite3
_cppyy              _testmultiphase_build html                sre_compile
_crypt              _thread             http                sre_constants
_csv                _threading_local    identity_dict       sre_parse
_ctypes             _tkinter            idlelib             ssl
_ctypes_test        _vmprof             imaplib             stackless
_ctypes_test_build  _warnings           imghdr              stat
_curses             _weakref            imp                 statistics
_curses_build       _weakrefset         importlib           string
_curses_cffi        _winapi             inspect             stringprep
_curses_panel       abc                 io                  struct
_dbm                aifc                ipaddress           subprocess
_decimal_build      antigravity         itertools           sunau
_ffi                argparse            json                symbol
_gdbm               array               keyword             symtable
_gdbm_build         ast                 lib2to3             sys
_gdbm_cffi          asynchat            linecache           sysconfig
_hashlib            asyncio             locale              syslog
_hpy_universal      asyncore            logging             tabnanny
_immutables_map     atexit              lzma                tarfile
_imp                audioop             macpath             telnetlib
_io                 base64              macurl2path         tempfile
_jitlog             bdb                 mailbox             termios
_locale             binascii            mailcap             test
_lsprof             binhex              marshal             textwrap
_lzma               bisect              math                this
_lzma_build         builtins            mimetypes           threading
_lzma_cffi          bz2                 mmap                time
_markupbase         cProfile            modulefinder        timeit
_marshal            calendar            msilib              tkinter
_md5                cffi                msvcrt              token
_minimal_curses     cgi                 multiprocessing     tokenize
_multibytecodec     cgitb               netrc               tputil
```

```
_multiprocessing    chunk               nntplib             trace
_opcode             cmath               ntpath              traceback
_operator           cmd                 nturl2path          tracemalloc
_osx_support        code                numbers             tty
_overlapped         codecs              opcode              turtle
_pickle_support     codeop              operator            turtledemo
_posixshmem         collections         optparse            types
_posixshmem_build   colorsys            os                  typing
_posixshmem_cffi    compileall          parser              unicodedata
_posixsubprocess    concurrent          pathlib             unittest
_pwdgrp_build       configparser        pdb                 urllib
_pwdgrp_cffi        contextlib          pickle              uu
_py_abc             contextvars         pickletools         uuid
_pydecimal          copy                pipes               venv
_pyio               copyreg             pkgutil             warnings
_pypy_generic_alias cpyext              platform            wave
_pypy_interact      crypt               plistlib            weakref
_pypy_irc_topic     csv                 poplib              webbrowser
_pypy_openssl       ctypes              posix               wsgiref
_pypy_testcapi      ctypes_support      posixpath           xdrlib
_pypy_util_build    curses              pprint              xml
_pypy_util_cffi     dataclasses         profile             xmlrpc
_pypy_util_cffi_inner datetime            pstats              zipapp
_pypy_wait          dbm                 pty                 zipfile
_pypy_winbase_build decimal             pwd                 zipimport
_pypy_winbase_cffi  difflib             py_compile          zlib
_pypy_winbase_cffi64 dis                 pyclbr              zoneinfo
_pypyjson           distutils           pydoc
_random             doctest             pydoc_data
```