2017 High School Programming Challenge Rules

Programming Challenge Computing Environment

The programming languages of the programming challenge will include C/C++ or Java. Each team will use a single workstation – the teacher's station at the front of the assigned room.

Each problem solution requires reading from a file.

Contestants of the challenge should only use the following editors/IDE:

- Eclipse (version Keplar SR 2)
- NetBeans IDE 8.0
- Microsoft Visual Studio 2013

Only official IDEs are permitted. Do not use the projector or any other equipment in the room.

Reference Materials

Teams may bring only written or printed notes on paper and books as reference. At the start of the programming challenge, each team will receive:

A green flash drive containing C++ and Java Docs. (This flash drive will also contain all the sample input files.)

Except for simple watches, teams may not have any electronic devices in their possession during the programming challenge. This includes, but is not limited to, flash drive, phones and other communication devices, calculators, calculator watches, smart watches, PDAs, electronic translators or dictionaries, e-books, audio players/recorders, video players/recorders, scanners, and printers. Mere possession of such a device will result in immediate disqualification, regardless of whether or not it was used.

Scoring of the Programming challenge

A problem is solved when the judges acknowledge it as accepted. The judges are solely responsible for accepting or rejecting submitted runs. Teams are ranked according to the most problems solved. For the purpose of awards, teams who solve the same number of problems are ranked by least total time. The total time is the sum of the time consumed for each problem solved. The time consumed for a solved problem is the time elapsed from the beginning of the programming challenge to the submittal of the accepted run plus 20 penalty minutes for every rejected run for that problem regardless of submittal time. There is no time consumed for a problem that is not solved. Programming Challenge Director will be able to specify any additional tie-breakers.

Accepted solutions will be sent back with an "accepted receipt" to acknowledge a correct solution to a problem.

No partial credit will be given.

Error Messages

If a submission is not correct it will be returned with the first applicable error message from the following list. A 20-minute time penalty will be assessed for each incorrect submission. The error messages are described below.

- Submission Error

This indicates a violation of the basic submission rules.
Submission errors include (but are not limited to)

- Compilation errors (warnings are OK)
- Submitting the wrong problem on the labeled flash drive (problem # does not match flash drive #)
- Submitting the executable instead of the source code (only .c, .cpp, or .java)
- Submitting a program consisting of more than one source file
- Using the wrong source or input file name
- Specifying your java source code to be a part of a named package
- Including a drive and/or path with a file name
- Prompting and waiting for keyboard input (see also Time Limit Exceeded).

- Run-time Error

The program crashed when run. Examples of run-time errors include (but are not limited to)

- Stack overflows
- I/O errors
- Heap overflows
- Invalid memory references (e.g., dereferencing a null pointer)
- Bizarre behavior (garbage on screen, beeping, keyboard lockup, ...).

- Time Limit Exceeded

If you get this error, the program ran (without crashing) for more than the time limit of 15 seconds. It may or may not have produced any output. This error may indicate that the program simply runs too slowly or that it was silently waiting for keyboard input, but it usually means that it got stuck in an infinite loop.

- Wrong Answer

The program ran to completion within the time limit but produced one or more incorrect answers. A program that produces unsorted output when the problem calls for sorted output will receive this error.

- Presentation Error

The program ran to completion within the time limit, produced answers that appear to contain essentially the same information as the correct answers, but formatted the output incorrectly in one or more cases. Examples of incorrect formatting include (but are not limited to)

- Misspelled words or omitted essential words
- Wrong case (e.g., uppercase instead of lowercase)
- Too few or too many spaces
- Wrong number of significant digits in numeric output
- Incorrect or missing punctuation
- Extra blank lines.
- Missing new line at the end of the final output line
- **Note:** Receiving a Presentation Error guarantees that there are one or more formatting errors, and indicates that the judges did not see any wrong answers based on a brief visual inspection. However, it is not a guarantee that the program produced no incorrect answers.

Materials Provided

Each team will be given:

- A set of problems
- A set of forms
    - Judging request for each problem. The form must accompany each submission.
    - A set of clarification request forms.
        * The request and its answers maybe automatically copied to every team
    - A set of print request forms. This form will be used to request a printed copy of any program.
- Directions for accessing the appropriate software on the programming challenge computer
- Paper notepads
- Pencils
- A green flash drive which has input files for each problem, Java Docs, and C++ Reference Library

Submissions

- Programs must be in a single source file with the name as indicated in the problem description; failure to meet this requirement is a Submission Error. Java programmers, note that you can include additional top-level classes in a single source file as long as they are unqualified (just "class", without "public"). Java programmers, note as well that your source file must **not** be declared as a part of a named package. That means you should **not** have a line at the top of your source code like: package someName;
- When submitting, save the source file (*.c, *.cpp, or *.java) onto the correctly labeled flash drive. Give the flash drive and green submission Judging Request form to the runner for judging. If your program is accepted as correct, you will receive a yellow Accepted Solution Receipt, otherwise the flash drive and the Judging Request form will be returned to you.
- All input must be from files provided on the green flash drive, as indicated in the problem's statement.
- All output must be exactly as shown in the examples. This applies to all problems, whether explicitly stated in the problem description or not. **Include a new line at the end of the last line of the output**. Spelling, punctuation, spacing, and case (uppercase/lowercase) are all significant.
- Programs must write their results to standard output (usually stdout in C, cout in C++, and System.out in Java). The judges will ignore all output to standard error (usually stderr in C, cerr/clog in C++, and System.err in Java), so you can write as much debugging information to standard error as you want, subject to time limit.
- Your program cannot require any intervention by the user. For example, if you pause the program and ask the user to press a key to continue, you will be flagged with a Time Limit Exceeded Error. If you pause the program without any prompting at all, you will also be flagged with a Time Limit Exceeded Error.
- Do not use drive and/or path specifications when naming input files. If a problem indicates that the input file is named file.in, then you must open file.in and not a:file.in or c:\stuff\file.in or anything else. Violating this rule will result in a Submission Error.
- All test cases used in judging will conform to the input specifications, as stated in the problem description. It is not necessary for you to detect invalid input.

- Input data and correct output data will obey the following rules.
    - Other than end-of-line characters, spaces are the only whitespace that appear.
    - Two or more consecutive spaces do not appear, unless specifically mentioned in the problem statement.
    - Spaces do not appear at the end of lines.
    - Spaces do not appear at the beginning of lines, unless specifically mentioned in the problem statement.
    - Blank lines do not appear, unless specifically mentioned in the problem statement.
    - All lines, including the last line, end with the standard end-of-line marker.

- This applies only to Java programmers. Counter to Java conventions, the name of your source file and main class must be in **lowercase** for this competition. For example, if a problem states that your program must be called compute, then you would create a file called compute.java that begins like this:

```
public class compute {
    public static void main(String args[]) {
    ...
    }
    ....
}
```

Other Rules

Any team that jeopardizes the integrity of the programming challenge or violates the rules of the programming challenge will be disqualified and the team members may be banned permanently from competing in the programming challenge. Some examples of such actions are:

- Accessing the Internet in **any** way,
- Disrupting power to computers,
- Corrupting judging materials or the judging process,
- Collaborating with anyone not on the team including your coach (this includes using any mobile device),
- Disobeying site officials' instructions regarding appropriate conduct.
- Using any non-approved electronic media
- Accessing non-approved software (Microsoft Word…, etc)
- Damaging site properties, such as computers, tables, chairs..,etc.