1st Annual Virginia Tech High School Programming Contest

We will follow the rules of the ACM ICPC (Intercollegiate Programming Competition) for this contest, which are reproduced here.

Online Setting

We are performing this contest in an online setting. Team may use the Internet for reference during the contest.

Teams may not

- Communicate with any human outside their team about the problems, including their coaches
- Use more than 1 computer during the contest

Teams may

- Use the Internet for reference
- Use a non-programmable calculator
- Ask their on-site coaches for assistance with technical matters, e.g. workstation internet access or computer setup.
- Use a printer (if available)
- Submit clarifications to the organizers (see Clarifications below).

On-site coaches are trusted with ensuring these conditions.

Problems

There are several problems to be solved, each numbered using a letter. These problems will be contained in a separate hand out.

The problems are NOT sorted by difficulty – figuring out how difficult a problem is is part of your task! In particular, do not assume that problem A is easiest.

One you have solved a problem, you should submit its source code via the online interface provided by PC^2's web client. A link will appear on http://icpc.cs.vt.edu. The submission will open for the practice & contest shortly before it starts.

All source code must be contained in a single source file.

Languages

The following programming environments are supported:

- Java JDK 1.8.0_20 (do not use a package statement)
- gcc/g++ 4.8.2
- Python 2.7.6
- Python 3.4.0

Judging

Solutions for problems submitted for judging are called runs. Each run will be judged. The judges will respond to your submission with one of the following responses. In the event that more than one response is applicable, the judges may respond with any of the applicable responses.

Response	Explanation
Yes	Your submission has been judged correct.
Wrong Answer	Your submission generated output that is not correct
Output Format Error	Your submission's output is not in the correct format or is misspelled
Incomplete Output	Your submission did not produce all of the required output
Excessive Output	Your submission generated output in addition to or instead of what is required
Compilation Error	Your submission failed to compile.
Run-Time Error	Your submission experienced a run-time error.
Time-Limit Exceeded	Your submission did not solve the judges' test data within 30 seconds.

Scoring

A team's score is based on the number of problems they solve and penalty points, which reflect the amount of time and number of incorrect submissions made before the problem is solved. For each problem solved correctly, penalty points are charged equal to the time at which the problem was solved plus 20 minutes for each incorrect submission. No penalty points are added for problems that are never solved. Teams are ranked first by the number of problems solved and then by the fewest penalty points.

Sample Input and Output

This problem set contains sample input and output for each problem, shown like this:

Sample Input

```
1 3 10
2 5 8
```

```
0 0 0
```

Sample Output

```
10.43
18.46
```

However, the judges will test your submission against longer and more complex datasets, which will not be revealed until after the contest. Your major challenge is designing other input sets for yourself so that you may fully test your program before submitting your run. Should you receive a judgment stating that your submission was incorrect, you should consider what other datasets you could design to further evaluate your program.

Clarifications

In the event that you feel a problem statement is ambiguous or incorrect, you may request a clarification. Read the problem carefully before requesting a clarification. If a clarification is issued during the contest, it will be broadcast to all teams. If the judges believe that the problem statement is sufficiently clear, you will receive the response, "No response, read problem statement." If you receive this response, you should read the problem description more carefully. If you still feel there is an ambiguity, you will have to be more specific or descriptive of the ambiguity you have found.

You may not submit clarification requests asking for the correct output for inputs that you provide, e.g., "What would the correct output be for the input ...?" Determining that is your job unless the problem description is truly ambiguous. Sample inputs may be useful in explaining the nature of a perceived ambiguity, e.g., "There is no statement about the desired order of outputs. Given the input: . . . , would both this: . . . and this: . . . be valid outputs?".

Processing Runs

Runs for each particular problem will be judged in the order they are received. However, it is possible that runs for different problems may be judged out of order. For example, you may submit a run for problem B followed by a run for problem C, but receive the response for C first.

Do not request clarifications on when a response will be returned. If you have not received a response for a run within 30 minutes of submitting it, please contact us to determine the cause of the delay. Under no circumstances should you ever submit a clarification request about a submission for which you have not received a judgment.

If, due to unforeseen circumstances, judging for one or more problems begins to lag more than 30 minutes behind submissions, a clarification announcement will be issued to all teams. This announcement will include a change to the 30 minute time period that teams are expected to wait before consulting the site judge.

Abuse Policy

The submission of abusive programs or clarification requests to the judges will be considered grounds for immediate disqualification.

The submission of code deliberately designed to delay, crash, or otherwise negatively affect the contest itself will be considered grounds for immediate disqualification.

Your Programs

All solutions must read from standard input and write to standard output. In C this is scanf/printf, in C++ this is cin/cout, and in Java this is System.in/System.out. In Python, it may be raw_input() or sys.stdin, or other functions. The judges will ignore all output sent to standard error (cerr in C++ or System.err in Java). You may wish to use standard error to output debugging information.

On your workstation you may test your program with an input file by redirecting input from a file:

```
./program < file.in
```

```
java Program < file.in
```

```
python program.py < file.in
```

General Output Rules

Unless otherwise specified, all lines of program output

- must be left justified, with no leading blank spaces prior to the first non-blank character on that line,
- must end with the appropriate line terminator (\n, endl, or println()), and
- must not contain any blank characters at the end of the line, between the final specified output and the line terminator.

You must not print extra lines of output, even if empty, that are not specifically required by the problem statement.

Number Formatting

Unless otherwise specified, all numbers in your output should begin with the minus sign (-) if negative, followed immediately by 1 or more decimal digits. If the number being printed is a floating point number, then the decimal point should appear, followed by the appropriate number of decimal digits. For output of real numbers, the number of digits after the decimal point will be specified in the problem description (as the “precision”).

All floating point numbers printed to a given precision should be rounded to the nearest value. For example, if 2 decimal digits of precision is requested, then 0.0152 would be printed as "0.02" but 0.0149 would be printed as "0.01".

In simpler terms, neither scientific notation nor commas will be used for numbers, and you should ensure that you use a printing technique that rounds to the appropriate precision.

Input Format and Validation

All input sets used by the judges will follow the input format specification found in the problem description. You **do not need to test** for input that violates the input format specified in the problem.

All lines of program input will end with the appropriate line terminator (e.g., a linefeed on Unix/Linux systems, a carriage return-linefeed pair on Windows systems).

If a problem specifies that an input is a floating point number, the input will be presented according to the rules stipulated above for output of real numbers, except that decimal points and the following digits may be omitted for numbers with no non-zero decimal portion.

Scientific notation will not be used in input sets unless a problem explicitly allows it.