2nd Annual Virginia Tech High School Programming Contest (2015)

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.

Teams <u>may not</u>

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

Teams <u>may</u>

- Use the Internet for reference (e.g. programming language documentation)
- Refer to any documentation prepared by themselves before the contest
- 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 shown online in the PCS contest management system and they also will be contained in a separate hand out.

The problems are NOT sorted by difficulty – figuring out how difficult a problem 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 PCS contest management system at https://icpc.cs.vt.edu/pcsvths. 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 1.8 - JDK 1.8.0_51 (do not use a `package` statement)
- C (C99 dialect, 64bit) using gcc 5.1.1 (`gcc -std=c99 -O2 –lm`)
- C++ (C++11 dialect, 64bit) using g++ 5.1.1 (`g++ -std=c++11 -O2 –lm`)
- Python 2.7.9
- Python 3.4.2

Programs are run using a standard installation of each of these packages, such as would be provided by a Fedora Docker container.

Judging

Solutions for problems submitted for judging are called runs. Each run will be judged by an automated judge program.

Response	Explanation
Correct	Your submission has been judged correct.
Wrong Answer	Your submission generated output that is not correct
Compiler Error	Your submission failed to compile.
Run-Time Error	Your submission experienced a run-time error.
Timeout	Your submission did not solve the judges' test data within the allotted time limit.

Your solution should match the required output character by character, but the PCS contest management system may decide to ignore insignificant whitespace or capitalization. For problems that involve floating point output, judging is based on an absolute and/or relative error which the problem specifies. Be sure to compute and print enough digits to achieve the desired precision.

All programs will be run on multiple test data input sets. The specification for each problem will specify bounds on the input data (e.g., how problem instances are contained in each test data input set). Some problems process only a single problem instance per run, some multiple. Be sure to carefully read the input specification. The timeout that is specified applies to a single test data input set (i.e., one run of your program). Timeouts are generally chosen based on a multiple (usually, 3x) of the slowest judge solution, rounded up to the nearest second.

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
```

Generally, the judging system will test your submission against larger and/or more complex datasets, which will not be revealed until after the contest. Your 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.

Clarifications will appear in a separate tab in the contest interface. Certain clarifications may be labeled “important” – those are also shown using a banner above the problem to which they apply.

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. Judging may be not be immediate, so do not wait for a result before you continue to work on the next problem! Judging may be delayed due to queuing or because the judge decide that a manual decision is required.

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.

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

Input Format and Validation

All input sets used by the judges will follow the input format specification found in the problem description. We ensure this by using automated validation tools. 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.