2009 Stanford Local ACM Programming Contest

Saturday, October 3rd, 2009

Read these guidelines carefully!

Rules

1. You may use resource materials such as books, manuals, and program listings. You may *not* search for solutions to specifics problems on the Internet, though you are permitted to use online language references (e.g., the Java API documentation) or algorithms references equivalent to what would be found in a standard algorithms textbook (e.g., CLRS). You may *not* use any machine-readable versions of software, data, or existing electronic code. That is, all programs submitted *must be typed during the contest*. No cutting and pasting of code is allowed.

2. You may not collaborate in any way (verbally, electronically, in writing, using gestures, telepathically, etc.) with other contestants, students, or anyone else during the contest.

3. You are expected to adhere to the honor code. You are still expected to conduct yourself according to the rules, even if you are not participating on site in the Gates building.

Guidelines for submitted programs

1. All programs must be written in C, C++, or Java. For judging, we will compile the programs in the following way:

   - `.c`: using `gcc -O2 -lm` (GCC version 4.2.4)
   - `.cc`: using `g++ -O2 -lm` (GCC version 4.2.4)
   - `.java`: using `javac` (Sun Java version 1.5.0)

   All programs will be compiled and tested on a Leland `myth` machine. The `myth` machines are dual Xeon 3.20 GHz machines with 2 GB RAM running Ubuntu Linux 7.10. Compilation errors or other errors due to incompatibility between your code and the `myth` machines will result in a submission being counted incorrect.

2. Make sure you `return 0;` in your `main()`; **any non-zero return values may be interpreted by the automatic judge as a runtime error**.

3. **Java users:** Please place your `public static void main()` function in a public class with the same name as the base filename for the problem. For example, a Java solution for the `test` program should be submitted in the file `test.java` and should contain a `main()` in `public class test`.

4. All solutions must be submitted as a single file.

5. All programs should accept their input on **stdin** and produce their output on **stdout**. They should be batch programs in the sense that they do not require human input other than what is piped into stdin.

6. Be sure to follow the output format described in the problem exactly. We will be judging programs based on a `diff` of your output with the correct solution, so your program's output must match the judge output **exactly** for you to receive credit for a problem. As a note, each line of an output file must end in a newline character, and there should be no trailing whitespace at the ends of lines.

How will the contest work?

1. If you chose to work remotely from a home computer, we recommend that you test out your account on the online contest system by submitting a solution for the test problem shown on the next page. We will do our best to set up the contest host to accept test problem submissions Saturday morning until approximately 1:45 pm.

2. For those who choose to participate onsite, from 1:00 to 1:45 pm, you should select a computer (or find a place to plug in your laptop), set up your workspace and complete a test problem. Space in Gates B02 and the PUP cluster is limited, and will be available on a first-come first-served basis. You may also choose to work directly on one of the `myth` machines in Gates B08, although technically we do not have that room reserved for the contest.

3. At 2:00 pm, the problems will be posted on the live contest page in PDF format, all registered participants will be sent an e-mail that the problems have been posted, and we will distribute paper copies of the problems to contestants competing in either Gates B02 or the PUP cluster.

4. For every run, your solution will be compiled, tested, and accepted or rejected for one of the following reasons: *compile error*, *run-time error*, *time limit exceeded*, *incorrect output*, or *presentation error*. In order to be accepted, your solution must match the judge output exactly (according to `diff`) on a set of hidden judge test cases, which will be revealed after the contest.

    - Source code for which the compiler returns errors (warnings are ok) will be judged as *compile error*.
    - A program which returns any non-zero error code will be judged as *run-time error*.
    - A program which exceeds the time allowed for any particular problem will be judged as *time-limit exceeded* (see below).
    - A program which fails a `diff -w -B` will be judged as *incorrect output*.
    - A program which passes a `diff -w -B` but fails a `diff` (i.e., output matches only when ignoring whitespace and blank lines) will be judged as *presentation error*.
    - A program which passes a `diff` and runs under the time constraints specified will be judged as *accepted*.

5. For each problem, the time allowed for a run (consisting of multiple test cases) will be 10 seconds total for all test cases. The number of test cases in a run may vary from 20 to 200 depending upon the problem, so be sure to write algorithmically efficient code!

6. You can view the status of each of your runs on the live online contest site. Please allow a few minutes for your submissions to be judged. The site also provides a live scoreboard for you to watch the progress of the contest as it unfolds.

7. At 6:00 pm, the contest will end. No more submissions will be accepted. Contestants will be ranked by the number of solved problems. Ties will be broken based on total time, which is the sum of the times for correct solutions; the time for a correct solution is equal to the number of minutes elapsed since 2:00 pm plus 20 penalty minutes per rejected solution. No penalty minutes are charged for a problem unless a correct solution is submitted. After a correct submission for a problem is received, all subsequent incorrect submissions for that problem do not count towards the total time.

8. The results of this contest will be used in part to select team members for representing Stanford at the forthcoming ACM regional competition. Six or more contestants have been customarily invited to the Stanford ACM ICPC teams in previous years.

Helpful hints

1. **Make sure your programs compile and run properly on the `myth` machines.** If you choose not to develop on the Leland systems, you are responsible for making sure that your code is portable.

2. **Read (or skim) through all of the problems at the beginning to find the ones that you can code quickly.** Finishing easy problems at the beginning of the contest is especially important as the time for each solved problem is measured from the beginning of the contest. Also, check the leaderboard frequently in order to see what problems other people have successfully solved in order to get an idea of which problems might be easy and which ones are likely hard.

3. If you are using C++ and unable to get your programs to compile/run properly, try adding the following line to your `.cshrc` file

   ```
   setenv LD_LIBRARY_PATH /usr/pubsw/lib
   ```

   and re-login.

4. The `myth` machines in Gates B08 are not officially reserved for the contest, but these will be the machines used for judging/testing of all programs. You may find it helpful to work on these machines in order to ensure compatibility of your code with the judging system.

5. If you are a CS major and have a working `xenon` account, please work in the **PUP cluster** rather than Gates B08; the PUP cluster has really nice Linux machines with very little load. (Do check the Java configuration if that affects you though!) If you don't know where the PUP cluster is, just ask!

6. If you are working on your own (Windows) laptop in Gates B02, but plan to ssh into a myth machine, it may be helpful to run a VNC session. Check out the IT services page on using VNC, which can be found at `http://unixdocs.stanford.edu/moreX.html`. If you wish to use an IDE (e.g., Visual Studio or Eclipse), please make sure that you know how to set this up yourself beforehand. We will not be able to provide technical support related to setting up IDEs during the contest.

7. If you need a clarification on a problem or have any other questions, post an clarification request to the live contest page, or just come talk to us. The contest judges will likely be hiding in the myth cluster, **Gates B08**.

The directions given here are originally based on those taken from Brian Cooper's 2001 Stanford Local Programming Contest problem set, and have been updated year after year to the best of our ability. The contest organizers would like to thank the problem authors of 2009, in alphabetical order, Andy Nguyen, Chuong Do, Jonathan Lee, and Sonny Chan.