2007 Stanford Local Programming Contest

Saturday, October 6, 2007

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. Students may not collaborate in any way with each other or anybody else, including people contacted via Internet.

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 in Gates B02.

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 4.0.3)
   - `.cc`: using `g++ -O2 -lm` (GCC version 4.0.3)
   - `.java`: using `javac` (Sun Java version 1.6.0)

   All programs will be compiled and tested on a Leland `myth` machine. The `myth` machines are dual Xeon 3.20 GHz machines with 1 GB RAM running Ubuntu Linux 6.06. Java programs will be invoked with the following command line:

   ```
   java -Xms512m -Xmx512m problemname
   ```

   which sets the amount of memory allocated for the program heap to 512 MB. Compilation errors or other errors due to incompatibility between your code and the `myth` machines will result in a submission being counted incorrect. The base filename for each problem will be listed in the problem statement; please use only the listed filename extensions.

2. Make sure you `return 0;` in your `main()`; **any non-zero return values will be interpreted by the automatic grader 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 the submission program to see if it works for you. To do this, **submit a solution for the test problem** shown on the next page. You may not see a result from the grading program immediately. However, you should receive a grading response by Friday midnight; if you do not, please let us know. We will also be grading test problems from 12:00 to 12:50 pm on Saturday.

2. For those who choose to participate onsite, from 12:00 to 12:50 pm, you will select a computer, 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 1:00 pm, the problems will be posted on the main contest page in PDF and PS 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 who choose to compete 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*, *wrong answer*, 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 *wrong answer*.
   - 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 write efficient code!

6. Watch your **e-mail on Leland**. We will be sending out any necessary messages as well as notification if your programs are accepted or rejected via e-mail, and the e-mails will go to `your_leland_id@stanford.edu`.

7. If you want to follow the progress of the contest, go to the **standings web page**. A link to this page exists on the main contest page, `http://ai.stanford.edu/~chuongdo/acm/2007/`.

8. At 5:00 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 1:00 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.

9. The top six contestants will advance to the regionals. Full results will be posted as soon as possible after the competition.

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 B02; the PUP cluster has UNIX machines, which may be a more convenient programming environment if you intend to use Emacs, etc. If you don't know where the PUP cluster is, just ask!

6. If you are working on a PC in Gates B02, it may be helpful to run a VNC session if you don't like coding from a terminal. 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, come talk to us in **Gates B02** or the **PUP cluster**, or send an e-mail to `chuongdo@cs`.

The directions given here are based on those taken from Brian Cooper's 2001 Stanford Local Programming Contest problem set. Many thanks to Sonny Chan, David Arthur, and Tomas Rokicki for their help in writing and debugging problems!

0 Test Problem (`test.{c,cc,java}`)

0.1 Description

This is a test problem to make sure you can compile and submit programs. Your program for this section will take as input a single number N and return the average of all integers from 1 to N, inclusive.

To submit a solution, you must log into an `myth` machine on the Leland network. Then, run the `/afs/ir.stanford.edu/users` program with a single argument: the name of the file you are submitting. For each problem, your solution must adhere to the naming convention specified next to the problem name! For example, to submit a C++ solution to this problem, type (make sure to run this from a `myth` machine):

```
$ ~chuongdo/acm/submit test.cc
```

When you submit your solution, the judge will receive an e-mail and will judge your solution as soon as possible (please be patient!). Once this is finished, you will receive an e-mail stating that you have completed the problem or that you have not, and a reason why not. You can resubmit rejected solutions as many times as you like (though incurring a 20 minute penalty for each rejected run of a problem you eventually get right). Once you have submitted a correct solution, future submissions of that problem will still be graded but will not count towards your final score or total time.

Note that you do not need to submit this problem during the actual contest!

0.2 Input

The input test file will contain multiple test cases. Each test case is specified on a single line containing an integer N, where $-100 \leq N \leq 100$. The end-of-file is marked by a test case with $N = -999$ and should not be processed. For example:

```
5
-5
-999
```

0.3 Output

The program should output a single line for each test case containing the average of the integers from 1 to N, inclusive. You should print numbers with exactly two decimal places. For example:

```
3.00
-2.00
```

0.4 Sample C Solution

```
#include <stdio.h>

int main() {
  int n;
  while (1) {
    scanf("%d", &n);
    if (n == -999) break;
    if (n > 0) printf("%.2lf\n", (double)(n * (n + 1) / 2) / n);
    else printf("%.2lf\n", (double)(1 + n * (1 - n) / 2) / (2 - n));
  }
  return 0;
}
```

0.5 Sample C++ Solution

```
#include <iostream>
#include <iomanip>
using namespace std;

int main() {
  cout << setprecision(2) << setiosflags(ios::fixed | ios::showpoint);
  while (true) {
    int n;
    cin >> n;
    if (n == -999) break;
    if (n > 0) cout << double(n * (n + 1) / 2) / n << endl;
    else cout << double(1 + n * (1 - n) / 2) / (2 - n) << endl;
  }
  return 0;
}
```

0.6 Sample Java Solution

```
import java.util.*;
import java.text.DecimalFormat;

public class test {
    public static void main(String[] args) {
        Scanner s = new Scanner(System.in);
        DecimalFormat fmt = new DecimalFormat("0.00");
        while (true) {
            int n = s.nextInt();
            if (n == -999) break;
            if (n > 0) System.out.println(fmt.format((double)(n * (n + 1) / 2) / n));
            else System.out.println(fmt.format((double)(1 + n * (1 - n) / 2) / (2 - n)));
        }
    }
}
```