2005 Stanford Local Programming Contest

Saturday, October 8, 2003

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 problems on the Internet, though you are permitted to use online language references (e.g., the Java API documentation). You may *not* use any machine-readable versions of software, data, or existing electronic code. 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, I will compile the programs in the following way:

   - `.c`: using `gcc -O2 -lm`
   - `.cc`: using `g++ -O2 -lm`
   - `.java`: using `javac`

   All programs will be compiled and tested on a Leland `elaine` machine. Compilation errors or other errors due to incompatibility between your code and the `elaine` 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`, should contain a `main()` in `public class test`, and will be run using the command `java 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 careful to follow the output format described in the problem. I will be judging programs based on a `diff` of your output with the correct solution. 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, you may want to test out the submission script 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 me know. I will also be grading problems from 12:00 to 12:50 on Saturday.

2. For those who choose to participate onsite, from 12:00 to 12:50, you will select a computer, set up your workspace and complete a test problem. Space in Gates B02 (14 Windows PCs) and the PUP cluster (14 Unix machines) is limited, and will be available on a first-come first-serve basis. If more computers are needed, any remaining contestants will be sent to Sweet Hall (many Solaris machines, including the `elaine` machines on which all testing will be done).

3. At 1:00, 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 I 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 approximately 50-100 judge test cases, which will be revealed after the contest. The time limit for a run is *10 seconds total for all test cases combined*, so write efficient code!

5. Watch your **e-mail on Leland**. I 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`.

6. 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`.

7. At 5:00 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 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 top six contestants will advance to the regionals, subject to the constraint of a maximum of one graduate student per three person team. 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 `elaine` machines.** If you choose not to develop on the Leland systems, you are responsible for making sure that your code is portable.

2. 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.

3. 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!

4. 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 CS 193D page on using VNC, which can be found at `http://www.stanford.edu/class/cs193d/vnc.html`.

5. **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.

6. If you need a clarification on a problem or have any other questions, come talk to me 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 David Arthur for contributing several of the problems used in this contest!

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 sum of all integers from 1 to N, inclusive.

You must submit your solutions via the Leland system and the submission script I have created. **You may develop your programs on whatever platform you want, but submission must be via Leland**. To submit a solution, run the `~chuongdo/acm/submit.pl` script 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:

```
$ ~chuongdo/acm/submit.pl 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.

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 sum of the integers from 1 to N, inclusive. For example:

```
15
-14
```

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 ("%d\n", n * (n + 1) / 2);
    else printf ("%d\n", 1 + n * (1 - n) / 2);
  }
  return 0;
}
```

0.5 Sample C++ Solution

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

int main(){
  while (true){
    int n;
    cin >> n;
    if (n == -999) break;
    if (n > 0) cout << n * (n + 1) / 2 << endl;
    else cout << 1 + n * (1 - n) / 2 << endl;
  }
  return 0;
}
```

0.6 Sample Java Solution

```
import java.io.*;

public class test {
    public static void main (String args[]){
        try {
            BufferedReader br = new BufferedReader (new InputStreamReader (System.in));
            while (true){
                int n = Integer.parseInt (br.readLine());
                if (n == -999) break;
                if (n > 0) System.out.println (n * (n + 1) / 2);
                else System.out.println (1 + n * (1 - n) / 2);
            }
        } catch (IOException e){
            e.printStackTrace();
        }
    }
}
```