Stanford Local Programming Contest

Oct 4, 2003 12 pm – 4pm

Please read these guidelines carefully!

Rules

1. You may use resource materials such as books, manuals and program listings. You **may not** use any machine-readable versions of software, data or existing electronic code. All software submitted during the contest *must* be typed during the contest.
2. You may not collaborate with each other, or others during the contest. This is an *individual* contest.
3. You are expected to abide by the Stanford Honor Code, even though you may not be in Gates B-02 or the Pup cluster.

Program submission guidelines

1. All programs must be written in C, C++ or Java.
2. **C/C++ Users**: If you wish to submit a solution for problem number `x`, your code should reside in a single file named `x.c` or `x.cc`, depending upon the language you are using. There are 5 problems numbered 1 thru 5 (and a dummy Problem numbered 0 for practice). A submitted program will be compiled as follows:

   `.c`: using `gcc -lm`

   `.cc`: using `g++ -lm`
3. **Java users: Please place your `public static void main()` function in `public class Main` so that your program can be run using the command `java Main`**. The submit script will copy all `.java` files from the current directory. So it is recommended that solutions to different problem numbers be in different directories. We will use `/usr/pubsw/apps/jdk-1.4.1/bin/javac` on Leland machines to compile your Java programs.
4. All programs should accept their input on `stdin` and produce their output on `stdout`. Any output to `stderr` will be ignored.
5. Be careful to follow the output format specified in any problem. Your program will be judged based on a `diff` of your output with the correct solutions to our test cases.

How will the contest work?

1. From 12:00 to 1:00, students will pick a computer, set up their workspace and complete a test problem (Problem No 0). The purpose of this dummy problem is to make sure that the `submit` script, described below, works for you.
2. At 1:00, the contestants will gather outside **Gates B02** and will be handed out problems. You then return to your workspace and begin solving problems.
3. If you wish to work remotely, and cannot make it to Gates B02, the problems will also be available off the contest webpage:

   `http://www.cs.stanford.edu/~manku/contest/2003.html`

   .

4. To submit a problem, run the following script, with the current directory being the one that contains your solutions:

   `~manku/contest/bin/submit`

   The script will prompt you to enter the problem number that you are submitting.

5. You may submit a problem as many times as you want (but note that there is a penalty for incorrect submissions – see below). For every submission, your solution will either be accepted or rejected for one of the following reasons: *Compilation error, Run-time error, Time limit exceeded, Wrong Answer, or Presentation Error*. The time limit for a run is **1 minute**.

6. Watch your email on Leland: Accept/reject notifications will be send by email to

   `your_leland_id@stanford.edu`

7. If you wish to follow the progress of the contest, go to the contest webpage and click on `Standings`.

8. At 4:00, the contest will end. No more submissions will be accepted.

9. The contestants will be ranked by the number of problems solved correctly. Ties will be broken on the basis of total time used for correct solutions. "Time used for a correct solution", as per ACM rules, equals the time elapsed since start of contest + 20 minutes per rejected solution for that problem. This means that you should submit a correct solution *as soon as possible*. Also note that incorrect submissions of problems that you eventually never managed to get accepted, do not count.

10. The top-6 contestants will advance to the regionals, subject to the constraint of a maximum of one graduate student per three-person team.

If you have any questions, send email to `manku@stanford.edu`.

Best of luck!

Problem 0: Palindromes

A palindrome is a string that reads the same right-to-left and left-to-right. Your goal is to write a program that will read a series of strings and output whether the string is a palindrome or not. Each string will consist only of digits 0 and 1.

Input will consist of a sequence of strings, one string per line.
Output will consist of one sentence per line, as shown below.

Input	Output
`0000`	`0000 is a Palindrome.`
`01000`	`01000 is not a Palindrome.`
`0`	`0 is a Palindrome.`
`0101`	`0101 is not a Palindrome.`
`010010`	`010010 is a Palindrome.`