MIT Programming Contest

Individual Round Logistics

September 28, 2003

In this 5-hour contest, you will be presented with 6 programming problems. Your goal is to solve as many tasks as you can by writing a correct and efficient (enough) C/C++/Java program for each.

During the first 4 hours of the contest, there will be a visible scoreboard. During the last hour, you will still receive feedback on your submissions, but these results will not appear on the scoreboard.

1 Pre-contest Setup

Before the contest starts, you need to register for the contest:

```
add pliang
touch /mit/pliang/acm/roster/$USER
```

Then create (a symbolic link to) your submission directory:

```
ln -s /mit/pliang/acm/submit/$USER submit
```

You will submit programs later by copying files into the `submit` directory you just created.

2 About the problems

When the contest starts, you will be given a set of programming problems to solve. You should write your solution in C, C++, or Java, compile it, *test* it, and then submit it. (Repeat if necessary.)

Every program you write should read all input from standard input and write all output to standard output. Your output should have no extra blanks, lines, or characters unless otherwise stated by the problem. Do not attempt to open files, open network ports, do any other type of sketchy stuff, or risk disqualification. Make sure your program terminates with exit code of 0.

A program is correct if it meets the following criteria:

- outputs the correct answer
- terminates normally within 30 seconds
- uses at most 64 MB of memory

3 Compilation

We will use the following compile line to compile your program; you might want to use the same line when you do your test runs to make sure that your program does not fail because of compiler incompatabilities.

To compile your C/C++/Java program for problem `N`:

```
add java                           # allows you to use the Sun Java compiler
gcc -o <probN> <probN>.c           # C
g++ -o <probN> <probN>.cc          # C++
javac <probN>.java                 # Java
```

4 Printing

During the contest, you may print your source code or input files. Execute the following command:

```
lpr <filename>
```

Please pick up your print-out immediately.

5 Submission

When you are confident that your program is correct, copy your source code (*not* your executable!) into the `submit` directory you set up before the contest.

For example, to submit your C solution of problem 1, type:

```
cp prob1.c submit
```

When our grading system is ready to grade your source code, it will move the source code out of the `submit` directory. You can check if your program is being graded by checking that your program has been removed from `submit`. After a while, you should get e-mail about the fate of your program. Do not wait for the e-mail: keep on working.

6 Scoring

Your score for the contest is exactly the number of problems you solved correctly. If two people solve the same number of problems, we use time as a tie-breaker. Your *time* on a problem that you solved is the time from the beginning of the contest until your correct submission[1] plus an additional 20-minute penalty for every incorrect submission you made. Your total time is simply the sum of the times of the individual problems you solved.

For example, suppose you made the following submissions:

[1] Please don't submit programs for problems you have already solved correctly.

0:10	problem 1 (incorrect)
0:30	problem 3 (correct)
0:35	problem 2 (incorrect)
0:45	problem 1 (correct)

Then your total time is $30+(45+20) = 95$ minutes. Note that no penalty will be charged if you submit an incorrect program and never get it right later (problem 2).

7 Regulations

In this contest, you are expected to work on your own with no collaboration with anyone else. You can consult C reference manuals that you bring with you into the contest and online man pages. You are not allowed to use any form of e-mail, online source code or machine-readable code: everything you submit you must type in yourself during the contest. Of course, you may use your own pens, pencils, and paper.

8 Sample Problem: Add

Write a program to add two numbers.

Input

There will be several lines of input. Each line of input contains two integers separated by a space.

The input data is terminated by a line that contains two zeros, and should not be processed.

Output

For each line of input, output the sum of the two integers.

Example

Sample Input

```
1 2
6 3
0 0
```

Sample Output

```
Sum 1: 3
Sum 2: 9
```