Programming for Interaction with a Server

To allow an expansion in the types of problems that can be presented, this contest introduces a more flexible way of interacting with a program. Besides the usual way of testing a program by redirecting a test file as standard input, some problems will require the program to receive its standard input from and send its standard output to a server that can vary the program's input based on its previous output. Since this requires some slight modifications in how the program writes its output, the problem description will always warn you when your program will be interacting with a server instead of just reading a file.

The required modification is always to write to standard output using an output method that you know is *unbuffered*, or always to *flush* standard output explicitly after writing to it. The point is that the server needs to see your output immediately after you write it, and this may not happen if it's being buffered somewhere. Note that the rules for the automatic flushing of buffers vary by what is being output to: the fact that you see the output on your terminal immediately after your program writes it doesn't mean that the same thing will happen when it's outputting to the server.

Here are examples of how to flush standard output:

C	`fputs("1\n",stdout); fflush(stdout);`
C++	`cout << "1\n" << flush;`
Java	`System.out.println("1"); System.out.flush();`

Problems that require communication with a server will provide a script that you can run to set up the communication with the server and to run your program with it.