Testing Plays It Safe at the UCF High School Programming Tournament

Reasons to Test Your Code:

Each problem in this contest will provide sample input and corresponding sample output that provide one or more examples of how a correct solution program will work. However, the judges will test your code much more exhaustively, using inputs that do not appear in these samples! It is highly worthwhile to craft your own test cases.

When the judges test your code, they will never type input at the keyboard, even though your program is reading from the standard input stream. Typing is too error-prone, and takes too long for large inputs. So it is also advisable to test your solution using a method similar to what the judges will use.

How the Judges Test:

The judges have text files which contain all their input test cases and corresponding outputs. The judges will always run your code, in its binary form, using *piped I/O* with these text files. For example, if you submit a file `problem.c` and the judging software compiles it to `problem.exe`, then the judges might run the program from a command prompt using syntax similar to the following:

```
problem.exe < problem_input.txt > team_output.txt
```

If you had submitted a Java solution, `problem.java`, instead, then the judges might use something like this:

```
java problem < problem_input.txt > team_output.txt
```

In either case, this runs your code using input from the file `problem_input.txt` and creates a text file `team_output.txt` with the output of your program (if there was already a file called `team_output.txt` in that folder, it will get replaced on each re-run of the program). Note that the specific filenames do not matter. The source code does not refer to any specific input file by name—it simply reads from the standard input stream.

Of course, the judges might use judging software rather than typing these exact commands directly into a command prompt window, though the results will be the same.

How to Test Your Code:

The following steps to test your code are recommended:

1. Become familiar with the basic functions of the command prompt on your system. For example, the commands `cd` and `dir` are useful for navigating folders and finding files.
2. Compile your code and use the command prompt to find the binary (compiled) file. This will show up in the command prompt directory listing as a `.exe` file for C/C++ and a `.class` file for Java.
3. Type the test cases for the problem into a text file. You can use your source code editor or Notepad for this. It does not matter what the file is named, but save it in the folder with the binary files.
4. Be sure to create some additional test inputs, beyond what is in the sample input!
5. Run your solution's binary form using the syntax shown above, with your input file specified after the < symbol, and use any name to create the output file after the > symbol.
6. Open the output file using your source code editor or Notepad, and check the outputs for accuracy.

Happy testing!