acm High School Programming Contest @  sponsored by: ebay

2015 University of Virginia High School Programming Contest

Welcome to the 2015 University of Virginia High School Programming Contest. Before you start the contest, please be aware of the following notes:

Rules

1. There is one (1) problem in this packet. Balloons will not be awarded for the practice contest, but will be awarded for the actual contest. The one problem is as follows:

Problem	Problem Name
A	Number Summation

2. Solutions for problems submitted for judging are called runs. Each run will be judged.

   The judges will respond to your submission with one of the following responses. In the event that more than one response is applicable, the judges may respond with any of the applicable responses.

Response	Explanation
Yes	Your submission has been judged correct.
No - Wrong Answer	Your submission generated output that is not correct or is incomplete.
No - Output Format Error	Your submission's output is not in the correct format or is misspelled.
No - Excessive Output	Your submission generated output in addition to or instead of what is required.
No - Compilation Error	Your submission failed to compile.
No - Run-Time Error	Your submission experienced a run-time error.
No - Time Limit Exceeded	Your submission did not terminate within one minute.

3. A team's score is based on the number of problems they solve and penalty minutes, which reflect the amount of time and number of incorrect submissions made before the problem is solved. For each problem solved correctly, penalty minutes are issued equal to the time at which the problem was solved plus 20 minutes for each incorrect submission. No penalty minutes are added for problems that are never solved. Teams are ranked first by the number of problems solved and then by the fewest penalty minutes.

4. This problem set contains sample input and output for each problem. However, the judges will test your submission against several other more complex datasets, which will not be revealed until after the contest. One challenge is designing other input sets for yourself so that you may fully test your program before submitting your run. Should you receive a "wrong answer" judgment, you should consider what other datasets you could design to further evaluate your program.

5. In the event that you think a problem statement is ambiguous or incorrect, you may request a clarification. Read the problem carefully before requesting a clarification. If the judges believe that the problem statement is sufficiently clear, you will receive the response, "The problem statement is sufficient; no clarification is necessary." If you receive this response, you should read the problem

description more carefully. If you still think there is an ambiguity, you will have to be more specific or descriptive of the ambiguity you have found. If the problem statement is ambiguous in specifying the correct output for a particular input, please include that input data in the clarification request.

You may not submit clarification requests asking for the correct output for inputs that you provide. Sample inputs may be useful in explaining the nature of a perceived ambiguity, e.g., "There is no statement about the desired order of outputs. Given the input: ..., would not both this: ... and this: ... be valid outputs?".

If a clarification that is issued during the contest applies to all the teams, it will be broadcast to everybody.

6. Runs for each particular problem will be judged in the order they are received. However, it is possible that runs for different problems may be judged out of order. For example, you may submit a run for B followed by a run for C, but receive the response for C first.

   Do not request clarifications on when a response will be returned. If you have not received a response for a run within 30 minutes of submitting it, **you may have a runner ask the site judge to determine the cause of the delay. Under no circumstances should you ever submit a clarification request about a submission for which you have not received a judgment.**

   If, due to unforeseen circumstances, judging for one or more problems begins to lag more than 30 minutes behind submissions, a clarification announcement will be issued to all teams. This announcement will include a change to the 30 minute time period that teams are expected to wait before consulting the site judge.

7. The submission of abusive programs or clarification requests to the judges will be considered grounds for immediate disqualification. This includes submitting dozens of runs within a short time period (say, within a minute or two).

Your Programs

8. All solutions must read from standard input and write to standard output. In C this is `scanf()` / `printf()`, in C++ this is `cin` / `cout`, and in Java this is `System.in` / `System.out`. The judges will ignore all output sent to standard error (`cerr` in C++ or `System.err` in Java). You may wish to use standard error to output debugging information. From your workstation you may test your program with an input file by redirecting input from a file:

   ```
   program < file.in
   ```

9. All lines of program input and output should end with a newline character (`\n`, `endl`, or `println()`).

10. All input sets used by the judges will follow the input format specification found in the problem description. You do not need to test for input that violates the input format specified in the problem.

11. Unless otherwise specified, all lines of program output should be left justified, with no leading blank spaces prior to the first non-blank character on that line.

12. Unless otherwise specified, all numbers in your output should begin with a '-' if negative, followed immediately by 1 or more decimal digits. If it is a real number, then the decimal point should be followed by as many decimal digits as can be printed. This means that for floating point values, use

standard printing techniques (cout and System.out.println). Unless otherwise noted, the judging will check your programs with 10^{-3} accuracy, so only consider the sample output up until that point.

In simpler terms, neither scientific notation nor commas will be used for numbers, and you should ensure you do not round or use a set precision unless otherwise specified in the problem statement.

13. If a problem specifies that an input is a floating point number, the input will be presented according to the rules stipulated above for output of real numbers, except that decimal points and the following digits may be omitted for numbers with no fractional component. Scientific notation will not be used in input sets unless a problem statement explicitly specifies it.

Good luck, and HAVE FUN!!!

A. Number Summation

Description

Can you count to 10? Sure, you can! But can you also *add* the numbers from 1 to 10?

Given an integer i, print the sum of the numbers between 1 and i, inclusive.

Input Format

The first line of the file will have a number, n, which is the number of test cases in this file.

Each line will have a single integer value on it. The integers will always be positive, and will fit in a standard `int` variable.

Output Format

For each test case, print the sum on its own line. The sum will fit in a standard `int` variable.

Sample Input

```
3
1
5
10
```

Sample Output

```
1
15
55
```