2010 Mid-Atlantic Regional Programming Contest
Practice Round

Welcome to the practice round for the 2010 Programming Contest. Before you start the contest, please be aware of the following notes:

The Contest

1. There is one (1) practice problem. Please submit solutions or request clarifications **for this problem only.** Unless you have a real question about the problem, please submit at most one clarification request, and at most two runs. It is important that everyone have a chance to see how the system works. Even if you do not solve the practice problem, you should submit once just to practice with the system.

2. After (or even before) completing the practice problem, please read all of the notes listed here. They are designed to help you solve the problems during the contest.

3. 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
Correct	Your submission has been judged correct.
Wrong Answer	Your submission generated output that is not correct or is incomplete.
Output Format Error	Your submission's output is not in the correct format or is misspelled.
Excessive Output	Your submission generated output in addition to or instead of what is required.
Compilation Error	Your submission failed to compile.
Run-Time Error	Your submission experienced a run-time error.
Time-Limit Exceeded	Your submission did not solve the judges' test data within 30 seconds.

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

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

6. In the event that you feel 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 feel 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 both this: ... and this: ... be valid outputs?".

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

7. Runs for each particular problem will be judged in the order they are received. However, it is possible that runs for different problems (during the actual contest) 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.

8. The submission of abusive programs or clarification requests to the judges will be considered grounds for immediate disqualification.

Your Programs

9. 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
```

10. Every effort has been made to ensure that the compilers and run-time environments used by the judges are as similar as possible to those that you will use in developing your code. With that said, some differences may exist. It is, in general, your responsibility to write your code in a portable manner compliant with the rules and standards of the programming language. You should not rely upon undocumented and non-standard behaviors.

    One place where differences are likely to arise is in the size of the various numeric types. Many problems will specify minimum and maximum values for numeric inputs and outputs. You should write your code with the understanding that, on the *judges'* machines:

    - A C++ `int`, a C++ `long`, and a Java `int` are all 32-bits wide.
    - A C++ `long long` and a Java `long` are 64-bits wide.
    - A `float` in both languages is a 32-bit value capable of holding 6-7 decimal digits, though many library functions will be less accurate.
    - A `double` in both languages is a 64-bit value capable of holding 15-16 decimal digits, though many library functions will be less accurate.

    The data types on your own machines may differ in size from these, but if you follow the guidelines above in choosing the types to hold your numbers, you can be assured that they will suffice to hold those values on the judges' machines.

11. All lines of program input will end with the appropriate line terminator (e.g., a linefeed on Unix/Linux systems, a carriage return-linefeed pair on Windows systems).

12. 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.

13. 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,
    - should end with the appropriate line terminator (`\n`, `endl`, or `println()`), and
    - should not contain any blank characters at the end of the line, between the final specified output and the line terminator.

    You should not print extra lines of output, even if empty, that are not specifically required by the problem statement.

14. Unless otherwise specified, all numbers in your output should begin with the − if negative, followed immediately by 1 or more decimal digits. If it is a real number, then the decimal point should appear, followed by the appropriate number of decimal digits. For output of real numbers, the number of digits after the decimal point will be specified in the problem description (as the "precision").

    All real numbers printed to a given precision should be rounded to the nearest value. Rounding should be carried out so that trailing digits of 5 of higher are rounded up, tr4ailing digits of 4 or less are rounded down. For example, if a precision of 2 decimal digits is requested, then 0.0152 would round to 0.02, but 0.0149 would round to 0.01.

    In simpler terms, neither scientific notation nor commas will be used for numbers, and you should ensure you use a printing technique that rounds to the appropriate precision.

15. 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 non-zero decimal portion. Scientific notation will not be used in input sets unless a problem explicitly allows it.

Good luck, and HAVE FUN!!!

Practice Problem: Lunacy

After several months struggling with a diet, Jack has become obsessed with the idea of weighing less. In an odd way, he finds it very comforting to think that, if he had simply had the luck to be born on a different planet, his weight could be considerably less.

Of course, the planets are far out of reach, but even the Earth's moon would yield a dramatic weight loss. Objects on the moon weight only 0.167 of their weight on Earth.

Input

Input consists of one or more lines, each containing a single floating point number denoting a weight (in pounds) on the Earth. The end of input is denoted by a negative floating point number.

Output

For each line of input data, your program should print a single line of the form

`Objects weighing X on Earth will weigh Y on the moon.`

where X is the weight from the input and Y is the corresponding weight on the moon. Both output numbers should be printed to a precision of 2 digits after the decimal point.

Example

Input:

```
100.0
12.0
0.12
120000.0
-1.0
```

Output:

```
Objects weighing 100.00 on Earth will weigh 16.70 on the moon.
Objects weighing 12.00 on Earth will weigh 2.00 on the moon.
Objects weighing 0.12 on Earth will weigh 0.02 on the moon.
Objects weighing 120000.00 on Earth will weigh 20040.00 on the moon.
```