ACM International Collegiate Programming Contest
2002 East Central Regional Practice Contest
Ashland University
University of Cincinnati
Western Michigan University
Sheridan University
November 8, 2002

Sponsored by IBM

Rules:

1. All questions require you to read the test data from standard input and write results to standard output. You cannot use files for input or output. Additional input and output specifications can be found in the General Information Sheet.
2. All programs will be re-compiled prior to testing with the judges' data.
3. Non-standard libraries cannot be used in your solutions. The Standard Template Library (STL) and C++ string libraries are allowed.
4. Programming style is not considered in this contest. You are free to code in whatever style you prefer. Documentation is not required.
5. All communication with the judges will be handled by the PC^2 environment.
6. The allowed programming languages are C, C++ and Java.
7. Judges' decisions are to be considered final. No cheating will be tolerated.
8. There are **three** questions to be completed in **one hour and twenty minutes**.

Problem A: Climbing Worm

An inch worm is at the bottom of a well n inches deep. It has enough energy to climb u inches every minute, but then has to rest a minute before climbing again. During the rest, it slips down d inches. The process of climbing and resting then repeats. How long before the worm climbs out of the well? We'll always count a portion of a minute as a whole minute and if the worm just reaches the top of the well at the end of its climbing, we'll assume the worm makes it out.

Input

There will be multiple problem instances. Each line will contain 3 positive integers n, u and d. These give the values mentioned in the paragraph above. Furthermore, you may assume $d < u$ and $n < 100$. A value of $n = 0$ indicates end of output.

Output

Each input instance should generate a single integer on a line, indicating the number of minutes it takes for the worm to climb out of the well.

Sample Input

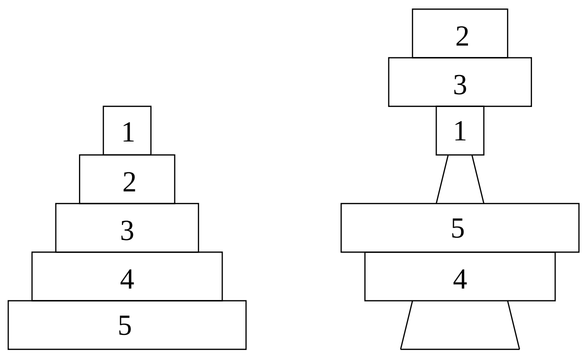
```
10 2 1
20 3 1
0 0 0
```

Sample Output

```
17
19
```

Problem B: Stacking Tower

One of the most common children's toys is a stacking tower, which consists of a series of rings of different sizes and a tapered rod which can hold the rings. The rings and rod are designed so that when the rings are placed in descending order by size, they fit exactly on the rod. Further, each ring, if placed by itself on the rod will go no lower than its position when all rings are on the rod. The diagram on the left shows an example of this toy which uses five rings, numbered 1 (smallest) to 5 (largest)



Unless endowed with super-genius mental faculties, most infants will place the rings in a random order on the rod, which results in some rings sticking over the top of the rod. The above diagram on the right shows the result of one such random placement. Your job will be to determine the number of rings which sit above the rod given a random ordering of the rings. You may assume that the rings may be stacked arbitrarily high without falling over.

Input

Input will consist of multiple problem instances. Each instance consists of a single line of the form $n\ r_1\ r_2\ \cdots\ r_n$. The positive integer n specifies the number of rings (≤ 100), and the remaining n integers give the order the rings are put on the rod. A value of $n = 0$ will terminate input.

Output

For each problem instance, you should output one line containing an integer indicating the number of rings sticking over the top of the rod.

Sample Input

```
5 5 4 3 2 1
5 4 5 1 3 2
8 1 2 3 4 5 6 7 8
20 11 10 19 7 12 14 5 2 3 1 8 6 13 17 18 9 4 20 15 16
0
```

Sample Output

```
0
2
7
11
```

Problem C: Best Fit

A multiple-geared bicycle typically has a gear in the front with two or three cogs, and a set of gears in the back, with five to nine cogs. In this problem, we will assume three cogs in the front and nine cogs in the back. The cyclist chooses a cog in the back and one in the front. The ratio (number of front teeth)/(number of back teeth) gives the relative difficulty for the gears. (The larger the ratio, the bigger the gear.) The gear setup is usually given in the form of a ratio. For example 42/21 indicates a 42-tooth cog in the front and a 21-tooth cog in the rear. How difficult it is to turn that gear also depends on the size of the rear wheel. When the ratio is multiplied by the rear wheel circumference, we get a measure of the size of the gear on that particular bike.

For example, a 27-inch diameter wheel has a circumference of 84.82293 inches (if we use the approximation of 3.14159 for π). A cyclist using a 52/15 setup would be riding a $(52/15) * 84.82293 = 294.052824$ inch gear. You'll be given the set of cogs on a bicycle and the diameter of the rear wheel. You'll also be given a target gear size and must then find the closest setup to that target.

Input

There will be multiple problem instances. The first line is a positive integer n indicating the number of problem instances to follow. Each of the next n lines will contain input for one problem instance. This line will consist of 14 positive integers in the form:

$f_1\ f_2\ f_3\ r_1 \cdots r_9$ *diameter target*

where $f_1 < f_2 < f_3$ are the three front cogs and $r_1 < r_2 < \cdots < r_9$ are the nine rear cogs, *diameter* is the diameter of the wheel and *target* is the target gear size.

Output

You should generate one line of output for each problem instance. This line should be of the form:

`A gear selection of` *ff*`/`*rr* `produces a gear size of` *size*`.`

where *size* is the closest computed gear size, rounded and displayed to three places, and *ff*/*rr* is the front cog/rear cog setup used for that gear size. (Use the approximation $\pi = 3.14159$ in your calculations.) If there is a tie for the closest size, use the one that uses the smallest front cog.

Separate outputs for problem instances with a blank line.

Sample Input

```
3
32 42 52 12 13 14 15 16 17 21 24 27 27 294
30 40 50 15 16 17 18 19 21 23 27 29 26 141
28 39 48 15 17 18 19 21 24 25 27 31 24 259
```

Sample Output

```
A gear selection of 52/15 produces a gear size of 294.053.

A gear selection of 50/29 produces a gear size of 140.830.

A gear selection of 48/15 produces a gear size of 241.274.
```