2010 Online Programming Contest
March 12, 2010

Sponsored by:
Dennis Matveyev

Rules:

1. There are six questions to be completed in 3 hours.
2. 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 document.
3. The allowed programming languages are C, C++, C# and Java.
4. All programs will be re-compiled prior to testing with the judges' data.
5. Non-standard libraries cannot be used in your solutions. The Standard Template Library (STL) and C++ string libraries are allowed. The standard Java API is available, except for those packages that are deemed dangerous by contestant officials (e.g., that might generate a security violation).
6. Programming style is not considered in this contest. You are free to code in whatever style you prefer. Documentation is not required.
7. All communication with the judges will be handled by the PC2 environment.
8. Judges' decisions are to be considered final. No cheating will be tolerated

A. Coins

This problem has additional scoring rules. Please see Scoring section below.

You are a manager of a fast food chain. Your registers are down again and you are the only person with a CIS degree. Given a dollar figure, you are requested to make change by telling your employees which coins to give back to customers. When giving change, use the least number of coins possible.

Input Specifications

Each line will have a decimal figure representing a dollar amount in a format of [D+.DD], where D is a digit 0 to 9, and + sign means that the previous digit repeats one or more times. Last line will have 0 and is not to be processed.

Output Specifications

For each input, output "Change for customer n:<EOL>dollars: D<EOL>quarters: r<EOL>dimes: d<EOL>nickels: n<EOL>pennies: p<EOL>", where n is the number of customer, starting at 1, and D, r, d, n, p are appropriate number of appropriate coin denominations. <EOL> is the end of line symbol. Separate each pair of consecutive customer output blocks by a blank line.

Scoring

In addition to the standard scoring rules, you will be charged 20 extra penalty points for solving this problem and for every problem you solve after solving this problem. This scoring will be applied post-match.

Sample Input

```
2.44
0.33
0
```

Sample Output

```
Change for customer 1:
dollars: 2
quarters: 1
dimes: 1
nickels: 1
pennies: 4

Change for customer 2:
dollars: 0
quarters: 1
dimes: 0
nickels: 1
pennies: 3
```

B. Finger Count

Look at your right hand. That's the one you type your code with! You should see a thumb and four fingers. Let's consider the thumb to be the first finger, numbered with a 1, as in the diagram below. Start counting from your first finger, until you reach number 5. Then, continue counting backwards through your fingers, until you reach your thumb again. Keep counting going forwards and backwards in this manner. Eventually you will get seriously bored.

To avoid boredom, here is a challenge: given a number, tell me which finger you will stop by counting to that number, and which finger you will visit next, if you were to continue counting.

Input Specifications

Each line will have a number between 1 and 2,147,483,647 inclusively. 0 will signify the end of input.

Output Specifications

Om Nom Nom has eated output specifications. See Sample Output instead.

Sample Input

```
3
9
14
0
```

Sample Output

```
Counting to 3 will stop on finger 3, going on 4.

Counting to 9 will stop on finger 1, going on 2.

Counting to 14 will stop on finger 4, going on 3.
```

C. Don't Be Afraid of the Function!

Student Anatolij Anton Antonovich is afraid of the Function. That evil professor Andreevich Andrej Adnreev is at it again! He gave his students an impossible assignment. He gave them … "The Function". Many stories exist at the school about "The Function". Students have cheated, failed exams, and broke up with their girlfriends, all because of "The Function". Rumors say some students even died… a little bit inside.

Now, for the first time, The Function can be seen in the day broadlight:

$$y = \left(\frac{x}{20}\right)^5 + 2\left(\frac{x}{20}\right)^4 - 52\left(\frac{x}{20}\right)^3 + 3\left(\frac{x}{20}\right)^2 + 524\frac{x}{20} - 329$$

I know this is hard, but be brave and do not look away. The Function will be afraid too.

The assignment is as such: given a monotonic range $[x_{low}, x_{high}]$ and a target value of y, find x within the range, such that $f(x) = y$

By monotonic range, the professor, of course, means that the function's first derivative in this range does not change sign.

Input Specifications

Each line of input will list three numbers, separated by spaces. These numbers will be x_{low}, x_{high}, and y, each of which fits in the IEEE's double value range. Input termination will be signified by three single-digit zeroes, separated by spaces. This last line containing zeroes is not to be processed.

Output Specifications

For each line of input, output a single number on a line by itself, formatted to four decimal places. This number must satisfy the constraints given in the problem. $f(x)$ will be considered to be equal to y if $|y - f(x)| \leq 1e - 11$. If per any chance, professor was as evil as to give an interval, where no value of x will give the desired target value of y, you are to output "Professor, there is no Function...". Bad things happen to students, when there is no Function. . . but don't let that scare you.

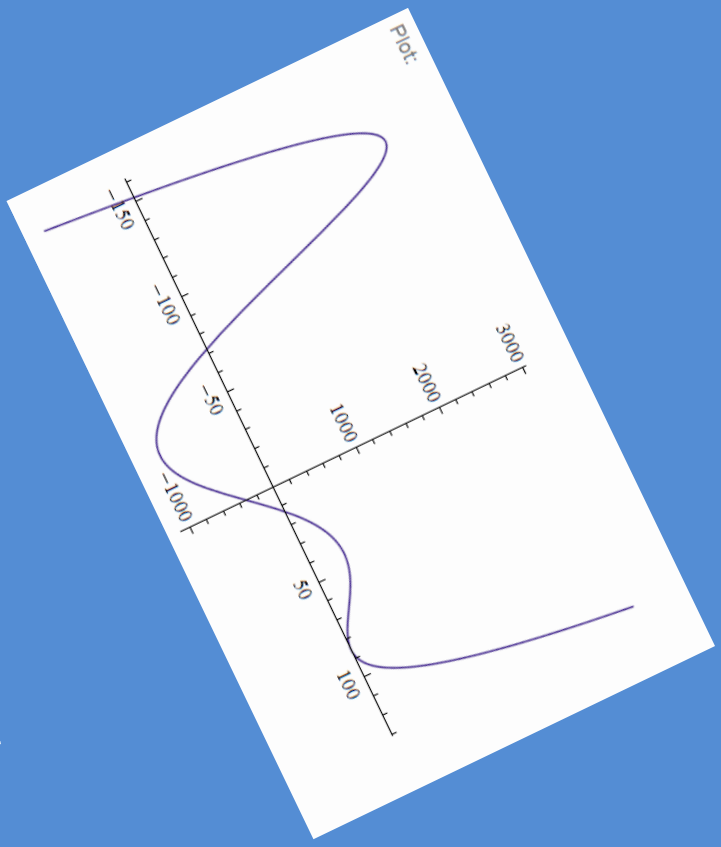
Sample Input

```
-36.1 43.1 3.1
-10 10 -100
-10 10 100
0 0 0
```

Sample Output

```
13.1740
8.8884
Professor, there is no Function...
```

Your Eyes Just Looked Here



OMG, The FUNCTION!

THIS SPACE HAS BEEN UNINTENTIONALLY LEFT DIRTY

D. Approximate This!

I am sure you've heard about π, the Greek letter for an extraordinary number. You may even know some approximations of π, such as 22/7 or 355/113. But are there better approximations? You are tasked with finding out the best approximation p/q to a given real number within given constraints. Best approximation is defined by the minimum absolute difference between the original number and your approximation.

Input Specifications

Each line of input will contain a representation of a number within the range of a positive double value between one and a million inclusively. Last line will contain 0 to signify the end of input.

Output Specifications

For each test case your program should output "Case n is best approximated as p/q", where n is the case number starting from 1 and p and q are positive integers, between 1 and 1,000,000 inclusively. Reduce fractions if needed. Separate each pair of consecutive output lines by a blank line.

Sample Input

```
3.141592653589793
2.718281828459045
1.111111111111111
0
```

Sample Output

```
Case 1 is best approximated as 833719/265381

Case 2 is best approximated as 566827/208524

Case 3 is best approximated as 10/9
```

E. Boxes

You are working for a box manufacturing company, in a multi-dimensional universe. You get a lot of orders to build all kinds of special boxes.

For example, an order would come in for a 2-dimensional box with sides 2 and 3. There are two possible configurations: length of 2 and width of 3, or length of 3 and width of 2. Since all boxes open at the "top", whatever that means in a multi-dimensional universe, there are just two unique boxes you could build for this order.

In another order a 3-dimensional box is to be built with sides 4, 4 and 4. No matter how you build this, there will only be one unique box with these sides.

You have been recruited by your boss to look at incoming orders in order to determine the number of unique boxes to be built.

Input Specifications

Input will start with a positive integer n, which is the number of dimensions to follow for a particular box order. The next n lines will be positive integers indicating the lengths of sides of the box to build. Last line with 0 dimensions is not to be processed.

Output Specifications

For each input, output a positive integer number indicating the number of unique boxes you could build for that order. All output will be less than or equal to $2^{31} - 1$.

Sample Input

```
2 2 3
3 4 4 4
4 5 2 2 5
0
```

Sample Output

```
2
1
6
```

F. Recover Sums

Vans Drussel is fond of his money. But he is also tired of long programming problems. So, instead of making up weird stories about money and how they relate to this problem, he'll just tell you what to do. Your task is as follows:

- Vans thinks up a source sequence of positive integer numbers such as {2, 3, 4, 6}.
- Vans then adds adjacent pairs together to yield a new sequence, in this case {5, 7, 10}.
- Vans then challenges you to recover the original sequence.

But there is a problem – there may be many different source sequences that generate the given sums. For example, {3, 2, 5, 5} is one such source sequence. Vans says he'll make this task easy – just find the source sequence that has the largest product. This product is formed by multiplying the members of the sequence together.

But wait! What if there is more than one sequence with the largest product? In this case, Vans says to find the sequence that's lexicographically first, meaning it has the least first number. If there is a tie for the first number, select the next least number and so on.

Now, for solving a problem like this, Vans is willing to pay good money.

Input Specifications

Each line will start with a number n, where $n \in \{1,2,3, \ldots ,100\}$. Following that there will be n numbers, each between 1 and 1,000. Last line will have 0 to signify end of input and is not to be processed.

Output Specifications

Find the source sequence that was used to generate a given sum sequence. All numbers in the source sequence must be positive integers greater than zero. Find the source sequence that has the largest product, and in case there is more than one such sequence, pick one that's lexicographically first. Output it in the form "Case c: n_1 n_2 … n_n", where c is the case number starting from 1, and n_1 n_2 … n_n is the sequence separated by spaces. If no source sequence exist, output "Case c: Impossible Sums!".

Sample Input

```
3 5 7 10
4 1 3 3 1
0
```

Sample Output

```
Case 1: 3 2 5 5
Case 2: Impossible Sums!
```