High School Programming Contest Problems
Fitchburg State University
May 3, 2013

You should write programs which will solve each of the following problems. For each problem we will give your program three sets of inputs and it should produce three sets of outputs; that is, the program will solve three instances of the problem, with different input each time.

Except where otherwise indicated, the set of inputs for each instance of the problem will be given on one line.

1. Factors

Every integer has a unique prime factoring; there are never two distinct multi-sets of prime numbers that multiply to the same integer (a multi-set is like a set but there may be identical members in a multi-set.) So the factors of twelve are the multiset {2 2 3}. This version of the definition of prime factoring is the reason that we don't count 1 as a prime, because if we did there would then be an infinite number of multisets of primes with the same product.

In this problem your program is given an integer, and it should print the smallest prime number which is a factor of the integer.

Sample Input

```
126
1681
262145
```

Output for the Sample Input

```
2
41
5
```

2. Parade

May Day -- the first of May-- is a holiday which seems to have disappeared from the American calendar. One version of May Day is the celebration of Labor Day, which is in September in the US but May 1 many other places. In Moscow, there is an annual May Day parade. I've never been to Moscow, but I've been to a number of parades other places. Usually there are lots of different groups in a parade, some with floats, some with bands, others just marching.

In a certain parade, there are *N* bands, and each is part of a different organization's presentation. If you were the parade organizer, you might want to avoid discordance by separating the bands as much as possible. Since positions at the front and the back are especially prestigious, you might have other reasons for placing groups there, but often a parade is sandwiched with two bands.

If there are *M* groups in the parade altogether, and *N* of them are bands, AND *M* were evenly divisible by *N-1* you could separate each band by *(M-N)/(N-1)* non-band groups. Of course, there's likely to be a remainder, and we'd like to spread it out so that the total distance between all bands is greatest, while still keeping the difference between the minimum and maximum distance at most one. I think that means that the remainder should be distributed over the separations at the middle of the parade. If it's necessary to decide between adding one of the remainder counts to two separations equally distant from the middle of the parade, we'll put it toward the front, figuring that at the beginning of the parade people are paying more attention.

I'll give you *M* and *N*, and you'll print out a list of all the separations. In the output, the band at the head of the parade is denoted by H, and all the other bands by B.

Sample Input

```
12 3
12 5
13 5
```

Output for the Sample Input

```
H 5 B 4 B
H 2 B 2 B 2 B 1 B
H 2 B 2 B 2 B 2 B
```

3. **Ribbons**
Before leaving May Day, here's a puzzle based on the Maypole. As preparation for posing the problem, I watched a couple of Maypole dances on YouTube. Most of the Maypole dances I watched featured British grade school kids holding ribbons, which are attached to the top of the pole. In one dance the kids alternated running circles around each other, resulting in an interesting woven pattern, working its way down from the top. I think that there's a puzzle there, but this one is based on a simpler step: One dance is based on the children going in one direction while their ribbons wind down around the pole; then they turn around and go in the other direction until the ribbons are unwound.

If the dancers are tall enough to hold the ribbons out at right angles from the pole so that the ribbons cover each other completely at each turn, and there's no vertical component in the wrapping, then the number of possible revolutions depends only on the length of the ribbons and the circumference of the pole. (We'll neglect the change in circumference from all those ribbons wrapped around it.)

At the other extreme, if the ribbons are wrapped in spirals around the pole in a single layer, so that no pole is showing under the ribbon, the number of revolutions depends on the length of the ribbon, the width of the ribbon, and the number of ribbons.

$b\tan\theta$

If there are three ribbons, they will wrap like this:

$\tan(\theta) = \frac{a}{b}$

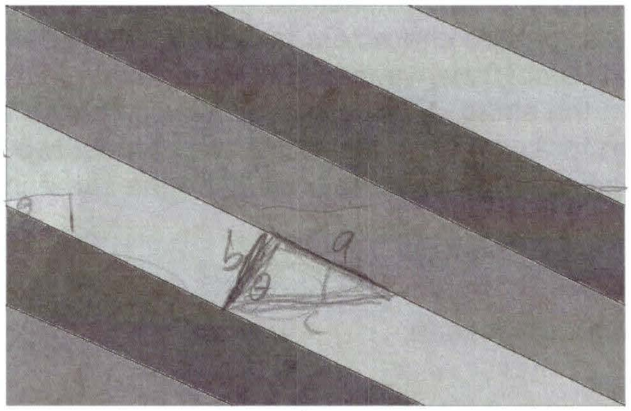
$\frac{1}{\cos\theta} = \frac{x}{b}$

$\cos\theta = \frac{b}{x}$

$\theta = \arccos\frac{b}{x}$

$\tan\theta = \frac{a}{b}$

$b = \text{width}$



where the picture unwraps a section of the surface of the pole into a rectangle. The width of the rectangle is the circumference of the pole, and the three bands in the middle are a single revolution of the dancers around the pole. Of course this single-layer covering is only possible if the total width of the ribbons is less than the circumference of the pole.

Your program will read in *C*, the circumference of the pole, *n* the number of ribbons, *W*, the width of the ribbons, and *L*, the length of the ribbons. Assuming that the height of the pole is sufficient, your program will print out two numbers: The maximum number of revolutions that the dancers can make around the pole (that would be assuming that the ribbons are applied one-on-top of another, and won't increase the circumference as they do so) and the number of revolutions they can make around the pole leaving no view of the pole between ribbons. Truncate your answers to an integer number of revolutions.

Sample Input

```
10 8 1 96
10 12 1 96
10 1 5 96
```

$\frac{C}{nw}$

w

Output for the Sample Input

```
7 9
6 9
8 9
```

C

4. **Distress**

Mayday is one of the international distress signals. It derives from the French phrase *m'aidez* meaning *help me*

The other well-known distress signal is the Morse code signal SOS, which consists of three dots, three dashes, and three dots. In this problem, we invite you to pick out SOS signals from a string, but instead of providing you with dots and dashes, we will use three repeated characters to denote each dash, one of the same characters to denote a dot, and we won't tell you ahead of time which character carries the signal in the string, although only one will in each string. Other characters are required for spacing, at least one and at most two between each dot or dash, but the spacing characters may not all be the same in a string.

For example, in the string

asafadfnansndnnnfnnngnnnhhnhnhhnasda

the SOS is signaled with an

n

although it looks at first as though the

a

might be a possibility.

For this problem, your program should figure out which, if any, character is signaling a SOS.

Sample Input

```
asafadfnansndnnnfnnngnnnhhnhnhhnasda
tttshshshshtsssesssasssasssaasbscsdqwery
bytherudebzrzizdgzzzezzz zzzarzczhzed the flood
```

Output for the Sample Input

```
SOS signaled with n
SOS not found
SOS signaled with z
```

5. Military time

There are a number of inconsistencies in the A.M. - P.M. time system in common use. For one thing, 11:59 AM is followed by 12:00 PM, and 11:59PM is followed by 12:00AM. A slightly more consistent scheme would be to explicitly label 12:00 Noon and 12:00 Midnight, but it seems bogus to speak of 12:01 Noon.

One solution to the inconsistencies is called *Military Time* or sometimes just 24-hour time. In this system, the clock runs from 00:01 to 24:00, and we dispense with the AM/PM distinction. For this problem, we'll give you a military time in the form 18:00 and your program will print out the time in the AM/PM system -- in this case 6:00 PM.

Sample Input

```
12:00
00:24
14:59
```

Output for the Sample Input

```
12:00PM
12:24AM
2:59PM
```

6. Rectangle

In how 'rectangular' a fashion can we place 1620 square solar panels?

Of course, we can place them 1 x 1620, 2 x 810, 18 x 90 ... We'd like, if possible, to have the ratio of the sides come as close as possible to the golden mean, (1+√5)/2, approximately 1.618033989 We assume that each panel occupies an approximately square space on the ground, and that we place them in rows of equal size, so that the ratio of the two factors of the total number of panels, is the ratio of the sides of the rectangle in which they are placed.

Your program will be given a number of panels, and will have to choose a best pair of factors. It should list the shorter of the two sides first.

Sample Input

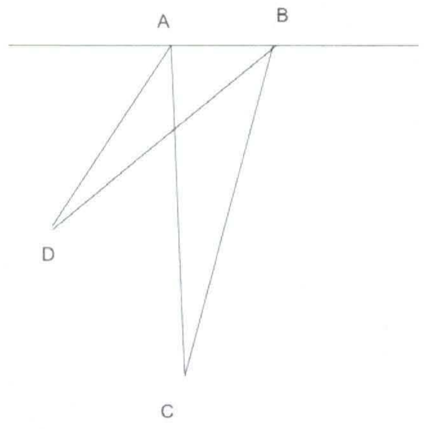
```
1620
1728
529
```

Output for the Sample Input

```
30 54
32 54
23 23
```

7. Window

Two friends, Charles and David, sit at the kitchen table, looking out the window. The question is, who can see more? The window appears in the diagram below as the line AB from the origin to (b,0) Charles is sitting at point C, with coordinates (cx, cy), and David is sitting at point D, with coordinates (dx, dy)



Although many different factors influence who can see more, including whether Charles is wearing his glasses and whether David is facing a brick wall, the only factor we will tease out in this problem is the size of the two angles ACB and ADB. Point A will always be the origin, and b is the length of the line AB.

Input to your program is the length b and the four coordinates cx, cy, dx, dy and output should be one of the three answers, Charles, David, Same.

Sample Input

```
30 -15 -15 15 -30
30 15 -2 15 -30
30 -15 -60 15 -60
```

Output for the Sample Input

```
David
Charles
David
```

8. **Dice**

In the February 2013 issue of Communications of the Association for Computing Machinery, Peter Winkler has a series of dice probability puzzles. As you may remember, Pascal drew heavily on dice examples when he developed the theory of probability. So you might have thought about somewhat similar problems before. Winkler asks: If six dice are rolled simultaneously, and N is the number of different numbers that occur, what is the probability that N is equal to four?

There are many different ways to approach this problem, but if you have a computer available, you can easily try all 46656 different outcomes of the problem. My attempt shows 23400 of the 46656 outcomes have four different numbers. If you don't have a computer available, or even after you have done the computer calculation, you can apply your mathematical intuition to see what makes sense.

I'm going to suggest a parameterized version of the problem. Suppose you have *s*-sided dice, you throw *m* of them, and you want to know what the probability that N is equal to some number *d*.

I'll give you *s*, *m*, and *d*, and your program will give me the probability, to three decimal places.

Sample Input

```
6 6 4
12 5 4
20 6 4
```

Output for the Sample Input

```
0.502
0.477
0.118
```

9. **Odometer**

Otto's auto odometer reached 100000 miles this week. He pulled over to the side of the freeway to celebrate, and while he was munching his candy bar, he noticed that in only one more mile, the number on the odometer would be a palindrome, that is, at 100001 it would read the same forward and backward. (Otto doesn't have a tenths field on his odometer.)

Which brings up a simple problem. Not every number is as beautiful as 100000, but every integer *x* is part of a sum *x+y* which is a palindrome, and for every integer *x* there is a smallest non-negative integer *y* for which *x+y* reads the same forward and backward.

For 100000 the smallest such integer is 1. For 123 it is 8. For 123000 it is 321. For 11 it is 0.

We will give your program an integer containing 9 or fewer digits, and it will print out the smallest integer which can be added to it to give a palindrome.

Sample Input

```
2000
2003
2198
```

Output for the Sample Input

```
2
109
24
```

10. **Fibonacci**

One of the many interesting properties of the Fibonacci series is that successive terms can be used for the place-values of a binary (but not base-two) numbering system.

The Fibonacci series is the series which begins:

1 1 2 3 5 8 13 21 34 55 89 144 ...

Each term of the series after the first two ones is the sum of the previous two elements of the series.

Using this scheme, one could represent the number six as 10001 (five plus one) 10010 (also five plus one) 1110 (three plus two plus one) or 1101 (also three plus two plus one.) For this problem we prefer the answer with the smallest value as a regular base-two number, that is, the representation with the ones furthest to the right, or six as 1101.

The first ten integers in this representation are 1, 11, 101, 111, 1011, 1101, 1111, 10101, 10111, 11011.

Sample Input

```
6
23
512
```

Output for the Sample Input

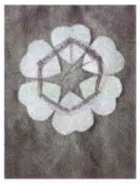
```
1101
1011011
1101111011111
```

11. **Heart**

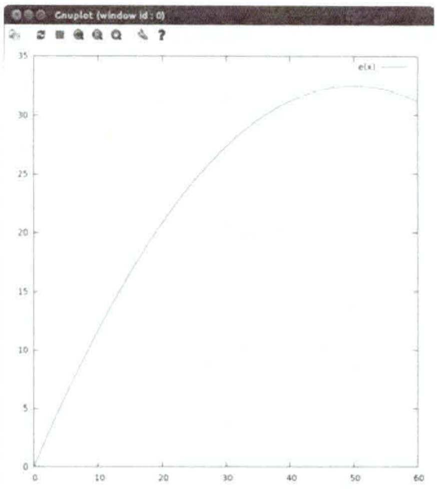
May Day used to be called May Basket Day. One might leave a May Basket on a neighbor's doorstep, then ring the bell and run so you weren't discovered. There was supposed to be a kiss involved if you got caught.

May baskets could contain flowers or candy. In my youth we favored those little hard candy hearts with sayings printed on them for May-Basket candy, possibly because they were more fun to play with than eat. In fact, playing with virtual candy hearts is the *point* of this problem.

A natural thing to do with candy hearts is arrange them in patterns. They make very nice shamrocks, four-leaf clovers, etc. Another neat puzzle is to try and cover an area with the least amount of area uncovered. In this problem, we arrange the hearts in a flower pattern, and attempt to compute how much space is left uncovered in the center of the pattern.

Here is an example with six hearts: 

In order to make the problem more computer friendly, I will provide you with the coefficients for a quadratic approximation to the lower left part of the heart. For example, here is a plot of the graph $h(x) = -0.013\, x^2 + 1.3\, x$



The nice thing about a quadratic is that we can easily differentiate and integrate it. If the quadratic expression is $h(x) = ax^2 + bx$, then the derivative with respect to x is $h'(x) = 2ax + b$, and the integral (or *anti-derivative*) is:

$$H(x) = ax^3/3 + bx^2/2$$

Then the slope of the line tangent to the curve $y = h(x)$ at any point x_1 is $h'(x_1)$, and the area bounded by the curve, the x axis, and the two vertical lines $y=x_1$ and $y=x_2$ is $H(x_2) - H(x_1)$

To give you a concrete example of the calculations involved (using the equation we have just seen above) in the six-heart flower, each heart takes up 60° of the circle, so each half takes up 30° which is a slope of 1/√3 or 0.57735, approximately.

If we arrange the hearts so that one of them is split in half by the positive x-axis (as our formula requires) then

- At the point that any two hearts touch (called the osculation point) a line drawn from the origin must be tangent to them both.
- And since that tangent has a slope of 1/√3, the derivative must have the value 0.57735 at that point, so we can solve

  $1.3 - 0.026\, x = 0.57735$

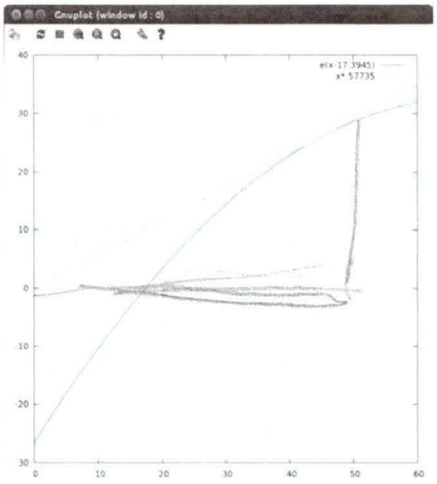
  to find the offset from the point of the heart which has the right slope. That's at x = 27.794.

- The height of the curve h(x) at that point is 26.0896.
- In order to find where a line from the origin with a slope of 0.57735 will pass through 26.0895, we solve the equation

$26.0896 = 0.57735\ x$

for x, finding $x = 45.1885$.

- So we need to offset the heart by 45.1885 - 27.794 = 17.3945 from the origin, as shown in the graph below.



- We need now only subtract the area of the heart to the left of the osculation point from the area of the triangle to the left of the same point, and multiply the result by 12.
  - Area of triangle = 45.1885 * 26.0895 / 2 = 589.472685375
  - Area of heart up to point where it touches:

    $[0.65\ x^2 - 0.00433333333333333\ x^3]^{27.794} = 409.0879932438766$

  - difference = 180.3846921311234
  - Area in center = 180.3846921311234 * 12 = 2164.616305573481

Your program will see inputs for *n*, the number of hearts to arrange in a flower pattern, *a* and *b*, the coefficients in the function

$ax^2 + bx$

and it will print out, to three decimal places, the area of the star in the middle of the flower which is not covered by the hearts.

Sample Input

```
-0.013  1.3   6
-0.013  1.3   4
-1 100 12
```

Output for the Sample Input

```
2164.654
65.237
277907829.801
```