Day 2 Question 1

High Tide

Input file: probd.in

Output file: probd.out

A planet has *n* moons revolving about it in constant clockwise coplanar circular orbits. How often do **all** the moons appear directly overhead as viewed from some point on the planet? We will call such a situation a 'vertical alignment'.

Your input consists of a number of sets of lines; each set consists of an integer *n*, indicating the number of moons, followed by *n* distinct positive integers, one per line, indicating the exact period of revolution for each moon, in days. (Thus there are *n*+1 lines of data for each set with the first line containing *n*.) The last line contains only a zero.

For each input set, except the last, generate a line of output indicating the interval in days, to two decimal places, between consecutive vertical alignments.

Sample Input

```
2
20
30
3
20
30
40
2
10
3
0
```

Sample Output

```
60.00
120.00
4.29
```

Day 2 Question 2

Space Aliens

Input file: probe.in

Output file: probe.out

Earth is being attacked by space aliens yet again. The invasion is being tracked by time-lapse photography. Three photographs have been taken at equal time intervals. Your job is to predict the positions of the alien craft in the next photograph, so that they may be intercepted by Earth's defence force. Each alien craft appears as a point of light in the plane of the photograph.

Your input consists of three snapshots of the crafts' positions, taken at 1/10 second intervals. During these intervals, each craft's speed and direction may be assumed to be constant. Your output consists of the coordinates at which the craft will appear 1/10 second after the third snapshot. (ie The positions in the fourth snapshot if there were one.)

Input

The input consists of several sets of data.
The first line in each set contains a positive integer n, the number of points of light in each picture. The next n pairs of lines of input contain the (x, y) coordinates of the points in the first picture, the x-coordinate on the first line and the y-coordinate on the second line. The next n pairs of lines contain the coordinates of the points in the second picture and the next n pairs of lines contain the coordinates of the points in the third picture. The points in each picture are in no particular order. The last line of the data file will contain only a zero.

Output

Your output is the set of coordinates where the points of light will be 1/10 second after the third picture. Print the number of points, followed by the (x, y) coordinates of each point. If there is more than one solution, print all solutions with a blank line separating each. If there is no solution, print "Impossible".

Sample Input

```
1.0
1.0
2.0
2.0
2.0
2.0
3.0
3.0
3.0
3.0
4.0
4.0
3
1.0
```

```
1.0
1.0
2.0
1.0
3.0
2.0
1.5
2.0
2.0
2.0
2.5
3.0
1.0
3.0
2.0
3.0
3.0
0
```

Sample Output

```
2
4.0 4.0
5.0 5.0
3
4.0 0.0
4.0 2.5
4.0 3.5

4.0 0.5
4.0 1.5
4.0 4.0
```

Day 2 Question 3

Aligning DNA

Input file: probf.in

Output file: probf.out

You are a biologist studying various patterns in DNA sequences. You need a program that will find regions of similarity in a pair of sequences, and align the similar regions by inserting spaces into the sequences.

A DNA sequence is represented as a string of characters from the set {A, C, G, T}.

For example, if you were given the sequences:

```
AATCGA
TCGAGG
```

your program would insert two spaces before the second sequence to align the TCGA. The aligned sequences would be:

```
AATCGA--
--TCGAGG
```

where the dashes represent spaces. In every alignment, the lengths of the two sequences (including spaces) must be equal, although the original sequences need not be of the same length.

We will assign a number of points to each alignment, and we will call the alignment with the highest number of points the best alignment. To calculate the number of points, we proceed character by character from the beginning of the sequence. For each position, we get a certain number of points:

- if one (or both) of the sequences contains a space at that position, we lose one point;
- if the two characters in a given position are different, we lose one point;
- if the two characters in a given position are the same, we get three points.

For example, with the alignment:

```
ACT
-GT
```

we would lose one point for the first position (A,-), another point for the second position (C,G) and gain three points for the last position (T,T), for a total of one point.

Your program will be given pairs of DNA sequences. It must find the best alignment, along with the number of points given to that alignment. If there is more than one alignment with the highest number of points, any one of the best alignments is acceptable.

The input will consist of pairs of lines, each line no longer than 100 characters. Each line represents a DNA sequence, and you are to align the two sequences in each pair of lines

(ie, align each odd-numbered line with the line that comes after it). The last line of the input will consist of a single asterisk. The input will not contain any lower-case letters.

The output is to consist of three lines for each pair of sequences. The first line will be a single integer representing the number of points assigned to that alignment. The next two lines will consist of the aligned sequences, in the same order as they were read in, with spaces represented by dashes (-). The output shall not contain any lower-case letters.