II MARATON INTERNA DE PROGRAMACION

UNIVERSIDAD KONRAD LORENZ

17 de Noviembre de 2007

PROBLEMAS

Elaborado por: Hector Florez
Basado de www.acis.org.co, www.acm.org

Problem 1

Interlines

Source file name: interlines.c, interlines.cpp or interlines.java
Input: interlines.in
Output: standar output

We all know that a pair of distinct points on a plane defines a line and that a pair of lines on a plane will intersect in one of three ways: [1] no intersection because they are parallel, [2] intersect in a line because they are on top of one another (i.e. they are the same line), [3] intersect in a point.
In this problem you will use your algebraic knowledge to create a program that determines how and where two lines intersect.

Input

Your program will repeatedly read in four points that define two lines in the x-y plane and determine how and where the lines intersect. All numbers required by this problem will be reasonable, say between -1000 and 1000.

The first line contains an integer N between 1 and 10 describing how many pairs of lines are represented. The next N lines will each contain eight integers. These integers represent the coordinates of four points on the plane in the order
x1 y1 x2 y2 x3 y3 x4 y4

Thus each of these input lines represents two lines on the plane: the line through (x1,y1) and (x2,y2) and the line through (x3,y3) and (x4,y4). The point (x1,y1) is always distinct from (x2,y2). Likewise with (x3,y3) and (x4,y4).

Output

There should be N+2 lines of output. The first line of output should read
INTERSECTING LINES OUTPUT

There will then be one line of output for each pair of planar lines represented by a line of input, describing how the lines intersect: none, line, or point. If the intersection is a point then your program should output the x and y coordinates of the point, correct to two decimal places. The final line of output should read
END OF OUTPUT

Sample input	Sample Output
5 0 0 4 4 0 4 4 0 5 0 7 6 1 0 2 3 5 0 7 6 3 -6 4 -3 2 0 2 27 1 5 18 5 0 3 4 0 1 2 2 5	INTERSECTING LINES OUTPUT POINT 2.00 2.00 NONE LINE POINT 2.00 5.00 POINT 1.07 2.20 END OF OUTPUT

Problem 2

Biggest Prime

Source file name: prime.c, prime.cpp or prime.java
Input: prime.in
Output: standar output

Definition

Prime number is which could be dividing only for 1 and for itself. You must find the biggest prime in a set of numbers

Input

There will be multiple cases. Each test case will be contained on one line. Each line will contain multiple integer numbers.

Output

For each test case you should print the biggest prime number. If one case does not contain any prime number you must print the word No.

Sample input	Sample Output
10 7 15 30 25 17 5 10 15 20 25 30 8 10 15 13 7	17 No 13

Problem 3

Speed Limit

Source file name: speed.c, speed.cpp or speed.java
Input: speed.in
Output: standar output

Bill and Ted are taking a road trip. But the odometer in their car is broken, so they don't know how many miles they have driven. Fortunately, Bill has a working stopwatch, so they can record their speed and the total time they have driven. Unfortunately, their record keeping strategy is a little odd, so they need help computing the total distance driven. You are to write a program to do this computation.
For example, if their log shows

Speed in miles per hour	Total elapsed time in hours
20	2
30	6
10	7

this means they drove 2 hours at 20 miles per hour, then 6-2=4 hours at 30 miles per hour, then 7-6=1 hour at 10 miles per hour. The distance driven is then (2)(20) + (4)(30) + (1)(10) = 40 + 120 + 10 = 170 miles. Note that the total elapsed time is always since the beginning of the trip, not since the previous entry in their log.

Input

The input consists of one or more data sets. Each set starts with a line containing an integer n, $1 \leq n \leq 10$, followed by n pairs of values, one pair per line. The first value in a pair, s, is the speed in miles per hour and the second value, t, is the total elapsed time. Both s and t are integers, $1 \leq s \leq 90$ and $1 \leq t \leq 12$. The values for t are always in strictly increasing order. A value of -1 for n signals the end of the input.

Output

For each input set, print the distance driven, followed by a space, followed by the word "miles".

Sample input	Sample Output
3 20 2 30 6 10 7 2 60 1 30 5 4 15 1 25 2 30 3 10 5 -1	170 miles 180 miles 90 miles

Problem 4

Symmetric Order

Source file name: order.c, order.cpp or order.java
Input: order.in
Output: standar output

In your job at Albatross Circus Management (yes, it's run by a bunch of clowns), you have just finished writing a program whose output is a list of names in nondescending order by length (so that each name is at least as long as the one preceding it). However, your boss does not like the way the output looks, and instead wants the output to appear more symmetric, with the shorter strings at the top and bottom and the longer strings in the middle. His rule is that each pair of names belongs on opposite ends of the list, and the first name in the pair is always in the top part of the list. In the first example set below, Bo and Pat are the first pair, Jean and Kevin the second pair, etc.

Input

The input consists of one or more sets of strings, followed by a final line containing only the value 0. Each set starts with a line containing an integer, n, which is the number of strings in the set, followed by n strings, one per line, sorted in nondescending order by length. None of the strings contain spaces. There is at least one and no more than 15 strings per set. Each string is at most 25 characters long.

Output

For each input set print "SET n" on a line, where n starts at 1, followed by the output set as shown in the sample output.

Sample input	Sample Output
7 Bo Pat Jean Kevin Claude William Marybeth 6 Jim Ben Zoe Joey Frederick Annabelle 0	SET 1 Bo Jean Claude Marybeth William Kevin Pat SET 2 Jim Zoe Frederick Annabelle Joey Ben

Problem 5

The next round number

Source file name: roundn.c, roundn.cpp or roundn.java
Input: roundn.in
Output: standar output

An N-digit *round number* is characterized as follows:

- It is an integer with exactly N digits, each of which is between 1 and 9, inclusively.
- The digits form a sequence with each digit telling where the next digit in the sequence occurs. This is done by giving the number of digits to the right of the digit where the next digit in the sequence occurs. If necessary, counting wraps around from the rightmost digit back to the leftmost.
- The leftmost digit in the number is the first digit in the sequence, and the sequence must return to this digit after all digits in the number have been used exactly once.
- No digit will appear more than once in the number.

For example, consider the number 81362. To verify that this is a round number, we use the steps shown below:

1. Start with the leftmost digit, 8: 8 1 3 6 2
2. Count 8 digits to the right, ending on 6 (note the wraparound): 8 1 3 6 2
3. Count 6 digits to the right, ending on 2: 8 1 3 6 2
4. Count 2 digits to the right, ending on 1. 8 1 3 6 2
5. Count 1 digit to the right, ending on 3. 8 1 3 6 2
6. Count 3 digits to the right, ending on 8, where we began. 8 1 3 6 2

Given a positive integer R, you must determine the smallest round number that is equal to or greater than R.

Input

You will be provided with one or more input lines, each with a single integer R having between 2 and 7 digits followed immediately by the end of line. The last line of the input will contain only the digit 0 in column 1.

Output

For each input number, determine the smallest round number that is equal to or greater than R. There will always be such a number for each of the input numbers. Display the resulting number in the format illustrated below.

Sample input	Sample Output
12 123 1234 81111 82222 83333 911111 7654321 0	Case 1: 13 Case 2: 147 Case 3: 1263 Case 4: 81236 Case 5: 83491 Case 6: 83491 Case 7: 913425 Case 8: 8124956

Problem 6

Rounders

Source file name: rounders.c, rounders.cpp or rounders.java
Input: rounders.in
Output: standar output

For a given number, if greater than ten, round it to the nearest ten, then (if that result is greater than 100) take the result and round it to the nearest hundred, then (if that result is greater than 1000) take that number and round it to the nearest thousand, and so on ...

Input

Input to this problem will begin with a line containing a single integer n indicating the number of integers to round. The next n lines each contain a single integer
$x (0 \le x \le 99999999)$.

Output

For each integer in the input, display the rounded integer on its own line.
Note: Round up on fives.

Sample input	Sample Output
9 15 14 4 5 99 12345678 44444445 1445 446	20 10 4 5 100 10000000 50000000 2000 500