ACM International Collegiate Programming Contest
Southwestern European Regional Contest
ETH Zurich, Switzerland, December 8, 1995

Practice Problem Set

Contents

Rules and Advice		Page 2
Problem A:	Inverse Tangens	Page 3
Problem B:	Word Counting	Page 4
Problem C:	Sorting	Page 5

Sponsored by Microsoft
Supported by Union Bank of Switzerland

Some Rules and Advice

All questions require you to read the test data from a single input file and to write the results to a single output file. The names of both files are given in the header of the problem. Your are not allowed to read or write any other files than the specified ones. Standard input and output are also considered files.

Output must correspond exactly to the provided sample output, including spelling and spacing.

All lines (including the last one) should be terminated with a new-line character, and no whitespace should appear at the end of a line unless explicitly specified. Tabs should never be used.

All programs will be re-compiled prior to testing with the judges' data. Non-standard libraries may not be used in your solutions. C programs may not include any files except: `ctype.h`, `math.h`, `stdio.h`, `stdlib.h`, `string.h`, and `strings.h`. Pascal programs may use the extended `Reset`, `Rewrite` and the `Close` statement, which are not part of the ISO Pascal standard.

After analyzing a program, the judges will send one of the following messages:

- *Program accepted*. Your program passed all tests and is accepted as correct.
- *Compile-time error*. The judges were not able to successfully compile your program. The compiler returned an error (not just a warning).
- *Run-time error*. Your program "crashed", i. e. it exited prematurely due to a run-time error.
- *Wrong answer*. The program run through one or more test cases without a run-time error but the output did not match the expected output.
- *Presentation error*. The output seems to be correct but it is not presented in the required format. Since it is not always easy to distinguish this message from the *wrong answer* message, it is only sent in obvious cases.
- *Time-limit exceeded*. Your program did not finish within the given amount of time.
- *Contest rule violation*. Your program violates a contest rule like calling non-standard libraries.

Programming style is not considered in this contest. You are free to code in whatever style you prefer.

The CPU time limit for all problems is 3 minutes except when specified otherwise.

All questions regarding the contest material should be submitted to the judges by filling in a *clarification request* form. You can ask the *runners* in the room for any non-contest related matters or for getting your printer output. The helpers will not answer any questions regarding the contest material.

Judges' decisions are to be considered final. No cheating will be tolerated.

Success!

Practice Problem A: Inverse Tangens

Source file: tangens.c / tangens.p
Input file: tangens.in
Output file: tangens.out

In 2-dimensional geometry, we often need to calculate the angle of a vector of order 2 (see figure 1).

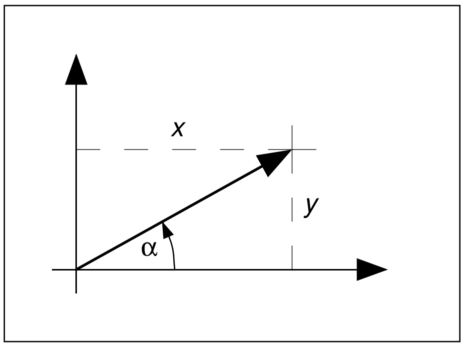

Figure 1: The angle α of vector (x, y)

The usual way to calculate the angle is to use the *arctan* function:

$$\alpha = \arctan \frac{y}{x}$$

Unfortunately, this does not work as expected. The *arctan* function always returns a value in the range $(-\pi/2, \pi/2)$. In particular, it returns the same value for the vectors (x, y) and $(-x, -y)$. We would prefer an *improved inverse tangens* function that returns an angle in the range $[0, 2\pi)$ and properly distinguishes between (x, y) and $(-x, -y)$.

You are to write a program that implements the *improved inverse tangens* function. It should read several vectors of order 2 from the input file, compute the angle of the vector using the *improved inverse tangens* function and write it to the output file.

Input

The first line of the input file contains the number of vectors to follow. Each following line contains one vector given as the x and y coordinate, separated by a blank.

Output

For each vector in the input file, your program should output the angle α of the vector as shown in figure 1. The floating-point numbers should include at least eight significant digits. For the vector (0, 0) you may output 0.

Example

Input

```
3
3.03 4.45
-4.99 3.39
-3.02 -4.43
```

Output

```
0.97300527
2.54485464
4.11404013
```

Practice Problem B: Word Count

Source file:	count.c / count.p
Input file:	count.in
Output file:	count.out

You are to write a program that counts the words in the input file. If the same word appears more than once, it should only be counted once.

Input

The input file contains several words. A word consists of lowercase and/or uppercase letters from A to Z. Words are delimited by one or more whitespace character (spaces, new-lines, tabs etc.). Other characters do not appear.

When comparing words, your program should be insensitive to case, i.e. „abc“ is the same as „ABC“ and „aBc“.

Output

Your output file should contain one integer number, giving the number of distinct words in the input file.

Example

Input

```
Sally sells sea shells
at   the   sea   shore
```

Output

```
7
```

Practice Problem C: Sorting

Source file:	sorting.c / sorting.p
Input file:	sorting.in
Output file:	sorting.out

You are to write a program that sorts several lists of integer numbers. Each list is to be sorted in descending order.

Input

The first line of the input file gives the number of lists to follow. Each list starts with a line giving the number n of elements in the list. The following n lines each contain one integer number.

Output

For each list found in the input file, your program should output the sorted list. The number n of elements should not be output. The numbers within a list should be in descending order.

Example

Input

```
2
3
1
2
3
4
4
6
8
10
```

Output

```
3
2
1
10
8
6
4
```