

2011-0913 UM Online Practice Programming Contest
September 13, 2011

Sponsored by:
University of Michigan

Rules:

1. There are a few questions to be tried out any time during the practice contest.
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 Sheet.
3. The allowed programming languages are 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. Bacteria

A number of bacteria lie on an infinite grid of cells, each bacterium in its own cell. Each second, the following transformations occur (all simultaneously):

1. If a bacterium has no neighbor to its north and no neighbor to its west, then it will die.
2. If a cell has no bacterium in it, but there are bacteria in the neighboring cells to the north and to the west, then a new bacterium will be born in that cell.

Upon examining the grid, you note that there are a positive, finite number of bacteria in one or more rectangular regions of cells. Determine how many seconds will pass before all the bacteria die. Here is an example of a grid that starts with 6 cells containing bacteria, and takes 6 seconds for all the bacteria to die. '1's represent cells with bacteria, and '0's represent cells without bacteria.

Initial State: 000010 011100 010000 010000 000000	1 second later: 000000 001110 011000 010000 000000	2 seconds later: 000000 000110 001100 011000 000000	3 seconds later: 000000 000010 000110 001100 000000
4 seconds later: 000000 000000 000010 000110 000000	5 seconds later: 000000 000000 000000 000010 000000	6 seconds later: 000000 000000 000000 000000 000000	

Input

The input consists of:

- One line containing **C** ($1 \leq C \leq 100$), the number of test cases.

Then for each test case:

- One line containing **R** ($1 \leq R \leq 10$), the number of rectangles of cells that initially contain bacteria.
- **R** lines containing four space-separated integers **X1 Y1 X2 Y2**:
($1 \leq X1 \leq X2 \leq 100$, $1 \leq Y1 \leq Y2 \leq 100$).

This indicates that all the cells with X coordinate between X1 and X2, inclusive, and Y coordinate between Y1 and Y2, inclusive, contain bacteria.

The rectangles may overlap. North is in the direction of decreasing Y coordinate. West is in the direction of decreasing X coordinate.

Output

For each test case, output one line containing "Case #N: T", where N is the case number starting from 1, and T is the number of seconds until all the bacteria die.

Sample Input

```
1
3
5 1 5 1
2 2 4 2
2 3 2 4
```

Sample Output

```
Case #1: 6
```

B. Play Game

You are playing a computer game and a big fight is planned between two armies. You and your computer opponent will line up your respective units in two rows, with each of your units facing exactly one of your opponent's units and vice versa. Then, each pair of units, who face each other will fight and the stronger one will be victorious, while the weaker one will be captured. If two opposing units are equally strong, your unit will lose and be captured. You know how the computer will arrange its units, and must decide how to line up yours. You want to maximize the sum of the strengths of your units that are not captured during the battle.

Input

There are several test cases. Each test case begins with an integer N ($1 \leq N \leq 50$) representing the number of units you and computer has. Then, the next line contains N integers representing the power of your units. There will be no white space before the first number and after the last number except the new line. Also, each integer is separated by one space. The last line also contains N integers representing the arrangement of units of computer with the same format above. Each unit has power range between 1 and 1,000, inclusive. The last test case ends with 0 and you should ignore this case.

Output

On the first line, you should print out the maximum total strength of your units that are not captured. Follow the format below.

Sample Input

```
5
5 15 100 1 5
5 15 100 1 5
0
```

Sample Output

```
120
```

C. An Excel-lent Problem

A certain spreadsheet program labels the columns of a spreadsheet using letters. Column 1 is labeled as "A", column 2 as "B", ..., column 26 as "Z". When the number of columns is greater than 26, another letter is used. For example, column 27 is "AA", column 28 is "AB" and column 52 is "AZ". It follows that column 53 would be "BA" and so on. Similarly, when column "ZZ" is reached, the next column would be "AAA", then "AAB" and so on.

The rows in the spreadsheet are labeled using the row number. Rows start at 1.

The designation for a particular cell within the spreadsheet is created by combining the column label with the row label. For example, the upper-left most cell would be "A1". The cell at column 55 row 23 would be "BC23".

You will write a program that converts numeric row and column values into the spreadsheet designation.

Input

Input consists of lines of the form: R_nC_m . n represents the row number and m represents the column number, $1 \leq n$, $m \leq 300000000$. The values n and m define a single cell on the spreadsheet. Input terminates with the line: R0C0 (that is, n and m are 0). There will be no leading zeroes or extra spaces in the input.

Output

For each line of input (except the terminating line), you will print out the spreadsheet designation for the specified cell as described above.

Sample Input

```
R1C1
R3C1
R1C3
R299999999C26
R52C52
R53C17576
R53C17602
R0C0
```

Sample Output

```
A1
A3
C1
Z299999999
AZ52
YYZ53
YZZ53
```

D. Minimum Rectangle Covering

Given N ($1 \leq N \leq 100$) points on the plane, find the rectangle of minimum area that covers them all.

Input

There are several data sets in the input text file. The first line of the input contains T , the number of test cases. Each case starts with N , the number of points. This is followed by N lines each containing two space separated integers x_i, y_i ($-1,000 \leq x_i, y_i \leq 1,000$).

Output

For each test case, output one number on a line by itself, the minimum area of a rectangle needed to cover all the points. Format your output to 6 decimal places, even if there are trailing 0s.

Sample Input

```
2
2
0 0
1 1
5
0 0
0 2
2 0
2 2
1 1
```

Sample Output

```
0.000000
4.000000
```

E. The Broken Record of DJ Zhzyatslya

DJ Zhzyatslya is in trouble. You see, after a successful career as a DJ, Zhzyatslya decided to try his hand at computing. But that did not work so well. After failing his Final Exam on History of Computing and Computation of Computable Things using Computers, DJ Zhzyatslya got very upset and smashed up his latest favorite one of a kind record collection. After he came back to sanity, he saw his life in pieces in front of him. Will you be so kind and help him put his records and his life back together?

By the way, DJ Zhzyatslya is very fond of his name. To ensure you pronounce it correctly, he provided you with IPA (International Phonetic Alphabet) transliteration: [z̥z̥ ɪ'a fʲs l̥ ɪ'a].

Input

Input will consist of multiple test cases. Each line will be in format $A / B + C / D$, where the letters represent numbers between 1 and 1000 inclusively. The last line will have all zeroes in place of the numbers. That line will not be processed.

Output

Sum up the fractions of DJ Zhzyatslya's Record Collection and print them out in a format A / B , if the sum is less than one. Use the format A and B / C , if the sum is larger than one. If the sum turns out to be a whole number, output only the whole part without the fraction, in a format of A . Always reduce the fractions to their lowest terms, when possible.

Sample Input

```
1 / 1 + 2 / 2
1 / 1 + 2 / 3
1 / 3 + 1 / 3
0 / 0 + 0 / 0
```

Sample Output

```
2
1 and 2 / 3
2 / 3
```