Colombian Collegiate Programming League
CCPL 2014

Contest 01 -- February 22

Problems

This set contains 10 problems; pages 1 to 15.

(Borrowed from several sources online.)

Official site `http://programmingleague.org`

Follow us on Twitter `@CCPL2003`

A - Add All

Source file name: `add.c`, `add.cpp`, *or* `add.java`

Yup!! The problem name reflects your task; just add a set of numbers. But you may feel yourselves condescended, to write a program just to add a set of numbers. Such a problem will simply question your erudition. So, let's add some flavor of ingenuity to it.

Addition operation requires cost now, and the cost is the summation of those two to be added. So, to add 1 and 10, you need a cost of 11. If you want to add 1, 2 and 3, then there are several ways:

1 + 2 = 3, cost is 3 3 + 3 = 6, cost is 6	1 + 3 = 4, cost is 4 2 + 4 = 6, cost is 6	2 + 3 = 5, cost is 5 1 + 5 = 6, cost is 6
Total is 9	Total is 10	Total is 11

I hope you have understood already your mission, to add a set of integers so that the cost is minimal.

Input

Each test case starts with a positive number $2 \leq N \leq 10^4$, followed by N positive integers each less than 10^5.

Input is terminated by a case where the value of N is zero. This case should not be processed.

The input must be read from standard input.

Output

For each case print the minimum total cost of addition in a single line.

The output must be written to standard output.

Sample Input	Sample Output
3 1 2 3 4 1 2 3 4 0	9 19

B - Brackets

Source file name: `brackets.c`, `brackets.cpp`, *or* `brackets.java`

Let us define a regular brackets sequence in the following way:

1. Empty sequence is a regular sequence.
2. If S is a regular sequence, then (S) and $[S]$ are both regular sequences.
3. If A and B are regular sequences, then AB is a regular sequence.

For example, all of the following sequences of characters are regular brackets sequences:

```
(), [], (()), ([]), ()[], ()[()]
```

And all of the following character sequences are not:

```
(, [, ), )(, ([)], ([(]
```

Some sequence of characters '(', ')', '[', and ']' is given. You are to find the shortest possible regular brackets sequence that contains the given character sequence as a subsequence. Here, a string $a_1a_2...a_n$ is called a subsequence of the string $b_1b_2...b_m$, if there exist such indices $1 \le i_1 < i_2 < ... < i_n \le m$, that $a_j = b_{i_j}$ for all $1 \le j \le n$.

Input

The input file contains several test cases, each test case is a string containing between 1 and 100 brackets (characters '(', ')', '[' and ']') that are situated on a single line without any other characters among them.

The input must be read from standard input.

Output

Output a line per test case that contains some regular brackets sequence that has the minimal possible length and contains the given sequence as a subsequence.

The output must be written to standard output.

Sample Input	**Sample Output**
`([(]`	`()[()]`

C - Roman Numerals

Source file name: `roman.c`, `roman.cpp`, *or* `roman.java`

A Roman numeral consists of a set of letters of the alphabet. Each letter has a particular value, as shown in the following table:

Letter	I	V	X	L	C	D	M
Value	1	5	10	50	100	500	1000

Generally, Roman numerals are written in descending order from left to right, and are added sequentially. However, certain combinations employ a subtractive principle. If a symbol of smaller value precedes a symbol of larger value, the smaller value is subtracted from the larger value, and the result is added to the total. This subtractive principle follows the next rules:

- `I` may only precede `V` and `X` (e.g., `IV` = 4).
- `X` may only precede `L` and `C` (e.g., `XC` = 900).
- `C` may only precede `D` and `M`.
- `V`, `L` and `D` are always followed by a symbol of smaller value, so they are always added to the total.

Symbols `I`, `X`, `C` and `M` cannot appear more than three consecutive times. Symbols `V`, `L` and `D` cannot appear more than once consecutively.

Roman numerals do not include the number zero, and for values greater or equal than 4000 they used bars placed above the letters to indicate multiplication by 1000.

You have to write a program that converts from Roman to Arabic numerals and vice versa. Although lower case letters were used in the Middle Ages, the Romans only used upper case letters. Therefore, for the Roman numerals we only consider upper case letters.

Input

The input consists of several lines, each one containing either an Arabic or a Roman number n, where $0 < n < 4000$.

The input must be read from standard input.

Output

For each input line, you must print a line with the converted number. If the number is Arabic, you must give it in Roman format. If the number is Roman, you must give it in Arabic format.

The output must be written to standard output.

Sample Input	**Sample Output**
XXV 4 942 MCMLXXXIII	25 IV CMXLII 1983

D - Forming Quiz Teams

Source file name: `teams.c`, `teams.cpp`, *or* `teams.java`

You have been given the job of forming the quiz teams for the next *MCA CPCI Quiz Championship*. There are $2 \cdot N$ students interested to participate and you have to form N teams, each team consisting of two members. Since the members have to practice together, all the students want their member's house as near as possible. Let x_1 be the distance between the houses of group 1, x_2 be the distance between the houses of group 2 and so on. You have to make sure the summation $x_1 + x_2 + x_3 + \cdots + x_N$ is minimized.

Input

There will be many cases in the input file. Each case starts with an integer N ($N \leq 8$). The next $2 \cdot N$ lines will given the information of the students. Each line starts with the student's name, followed by the x coordinate and then the y coordinate. Both x and y are integers in the range 0 to 1000. Students name will consist of lowercase letters only and the length will be at most 20.

Input is terminated by a case where N is equal to 0.

The input must be read from standard input.

Output

For each case, output the case number followed by the summation of the distances, rounded to 2 decimal places. Follow the sample for exact format.

The output must be written to standard output.

Sample Input	**Sample Output**
5 sohel 10 10 mahmud 20 10 sanny 5 5 prince 1 1 per 120 3 mf 6 6 kugel 50 60 joey 3 24 limon 6 9 manzoor 0 0 1 derek 9 9 jimmy 10 10 0	Case 1: 118.40 Case 2: 1.41

E - Tic Tac Toe

Source file name: `ttt.c`, `ttt.cpp`, *or* `ttt.java`

Tic Tac Toe is a child's game played on a 3 by 3 grid. One player, *X*, starts by placing an *X* at an unoccupied grid position. Then the other player, *O*, places an *O* at an unoccupied grid position. Play alternates between *X* and *O* until the grid is filled or one player's symbols occupy an entire line (vertical, horizontal, or diagonal) in the grid.

We will denote the initial empty Tic Tac Toe grid with nine dots. Whenever *X* or *O* plays we fill in an *X* or an *O* in the appropriate position. The example below illustrates each grid configuration from the beginning to the end of a game in which *X* wins.

```
...   X..   X.O   X.O   X.O   X.O   X.O   X.O
...   ...   ...   ...   .O.   .O.   OO.   OO.
...   ...   ...   ..X   ..X   X.X   X.X   XXX
```

Your job is to read a grid and to determine whether or not it could possibly be part of a valid Tic Tac Toe game. That is, is there a series of plays that can yield this grid somewhere between the start and end of the game?

Input

The first line of input contains *N*, the number of test cases. $4N - 1$ lines follow, specifying *N* grid configurations separated by empty lines.

The input must be read from standard input.

Output

For each case print `yes` or `no` on a line by itself, indicating whether or not the configuration could be part of a Tic Tac Toe game.

The output must be written to standard output.

Sample Input	Sample Output
2 X.O OO. XXX O.X XX. OOO	yes no

F - Flip Sort

Source file name: `flip.c`, `flip.cpp`, *or* `flip.java`

Sorting in computer science is an important part. Almost every problem can be solved effeciently if sorted data are found. There are some excellent sorting algorithm which has already acheived the lower bound nlgn. In this problem we will also discuss about a new sorting approach. In this approach only one operation (Flip) is available and that is you can exchange two adjacent terms. If you think a while, you will see that it is always possible to sort a set of numbers in this way.

A set of integers will be given. Now using the above approach we want to sort the numbers in ascending order. You have to find out the minimum number of flips required. Such as to sort ''`1 2 3`'' we need no flip operation whether to sort ''`2 3 1`'' we need at least 2 flip operations.

Input

The input will start with a positive integer N ($N \leq 1000$). In next few lines there will be N integers. Input will be terminated by `EOF`.

The input must be read from standard input.

Output

For each data set print

```
Minimum exchange operations : M
```

where M is the minimum flip operations required to perform sorting. Use a seperate line for each case.

The output must be written to standard output.

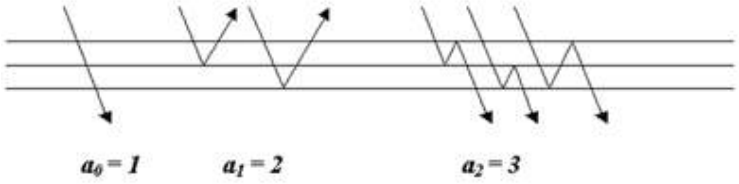
Sample Input	Sample Output
3 1 2 3 3 2 3 1	Minimum exchange operations : 0 Minimum exchange operations : 2

G - Ray Through Glasses

Source file name: `glasses.c`, `glasses.cpp`, *or* `glasses.java`

Suppose we put two panes of glass back-to-back. How many ways a_n are there for light rays to pass through or be reflected after changing direction n times?

The following figure shows the situations when the value of n is 0, 1 and 2:



Input

It is a set of lines with an integer n where $0 \leq n \leq 1000$ in each of them.

The input must be read from standard input.

Output

For every one of these integers a line containing a_n as described above.

The output must be written to standard output.

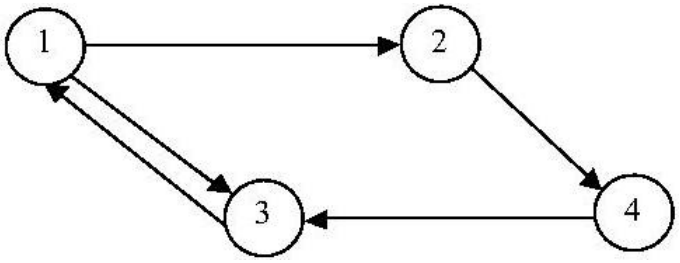
Sample Input	Sample Output
0 1 2	1 2 3

H - Page Hopping

Source file name: `hopping.c`, `hopping.cpp`, *or* `hopping.java`

It was recently reported that, on the average, only 19 clicks are necessary to move from any page on the World Wide Web to any other page. That is, if the pages on the web are viewed as nodes in a graph, then the average path length between arbitrary pairs of nodes in the graph is 19.

Given a graph in which all nodes can be reached from any starting point, your job is to find the average shortest path length between arbitrary pairs of nodes. For example, consider the following graph. Note that links are shown as directed edges, since a link from page *a* to page *b* does not imply a link from page *b* to page *a*.



The length of the shortest path from node 1 to nodes 2, 3, and 4 is 1, 1, and 2 respectively. From node 2 to nodes 1, 3 and 4, the shortest paths have lengths of 3, 2, and 1. From node 3 to nodes 1, 2, and 4, the shortest paths have lengths of 1, 2, and 3. Finally, from node 4 to nodes 1, 2, and 3 the shortest paths have lengths of 2, 3, and 1. The sum of these path lengths is $1 + 1 + 2 + 3 + 2 + 1 + 1 + 2 + 3 + 2 + 3 + 1 = 22$. Since there are 12 possible pairs of nodes to consider, we obtain an average path length of 22/12, or 1.833 (accurate to three fractional digits).

Input

The input data will contain multiple test cases. Each test case is defined by a sinle line containing an arbitrary number of blank-separated pairs of integers *a* y *b*, each representing a link from page numbered *a* to page numbered *b*. Page numbers will always be in the range 1 to 100. The input for each test case will be terminated with $a = b = 0$, which are not to be treated as page numbers. You can assume that the graph will not include self-referential links (that is, there will be no direct link from a node to itself), and at least one path will exist from each node in the graph to every other node in the graph.

The end of the input is given by line containing a pair of zeroes.

The input must be read from standard input.

Output

For each test case, determine the average shortest path length between every pair of nodes, accurate and rounded to three fractional digits. Display this length and the test case identifier (they're numbered sequentially starting with 1) in a form similar to that shown in the sample output below.

The output must be written to standard output.

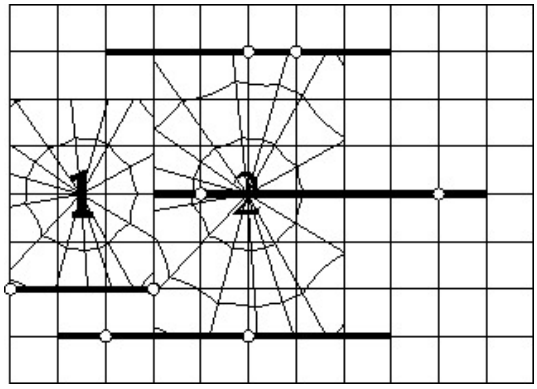
Sample Input	**Sample Output**
1 2 2 4 1 3 3 1 4 3 0 0 1 2 1 4 4 2 2 7 7 1 0 0 0 0	Case 1: average length between pages = 1.833 clicks Case 2: average length between pages = 1.750 clicks

I - Library

Source file name: `library.c`, `library.cpp`, *or* `library.java`

Castaway Robinson Crusoe is living alone on a remote island. One day a ship carrying a royal library has wrecked nearby. Usually Robinson brings any useful stuff from the shipwreck to his island, and this time he has brought a big chest with books.

Robinson has decided to build a bookcase for these books to create his own library. He cut a rectangular niche in the rock for that purpose, hammered in wooden pegs, and placed wooden planks on every pair of pegs that have the same height, so that all planks are situated horizontally and suit to act as shelves.



Unfortunately, Robinson has discovered that one especially old and big tome does not fit in his bookcase. He measured the height and width of this tome and has decided to redesign his bookcase in such a way, as to completely fit the tome on one of the shelves, taking into account locations of other shelves and the dimensions of the niche. With each shelf in the bookcase, one of the following operations should be made:

1. Leave the shelf on its original place.
2. Move the shelf to the left or to the right.
3. Shorten the shelf by cutting off a part of the plank and optionally move it to the left or to the right.
4. Move one of the pegs to a different place at the same height and move the shelf to the left or to the right.
5. Shorten the shelf by cutting off a part of the plank, move one of the pegs to a different place at the same height, and optionally move the shortened shelf to the left or to the right.
6. Remove the shelf from the bookcase along with both supporting pegs.

We say that the shelf is properly supported by its pegs, if exactly two distinct pegs support the shelf and the center of the shelf is between its pegs or coincides with one of the pegs. The original design of Robinson's library has all the shelves properly supported by their pegs and lengths of all shelves are integer number of inches. The Robinson may only cut an integer

number of inches from the planks, because he has no tools for more precise measurements. All remaining shelves after the redesign must be properly supported by their pegs.

You are to find the way to redesign Robinson's library to fit the special old tome without changing original design too much. You have to minimize the number of pegs that are to be removed from their original places during the redesign (operations 4 and 5 remove one peg, and operation 6 removes two pegs). If there are different ways to solve the problem, then you are to find the one that minimizes the total length of planks that are to be cut off (operations 3 and 5 involve cutting something from the planks, and operation 6 counts as if cutting off the whole plank). Width of planks and diameter of pegs shall be considered zero.

The tome may not be rotated. The tome should completely (to all its width) stand on one of the shelves and may only touch other shelves, their pegs or niche's edge.

Input

The input begins with a single positive integer on a line by itself indicating the number of the cases following, each of them as described below. This line is followed by a blank line, and there is also a blank line between two consecutive inputs.

The first line of the input file contains four integer numbers X_N, Y_N, X_T, and Y_T, separated by spaces. They are, correspondingly, width and height of the niche, and width and height of the old tome in inches ($1 \leq X_N, Y_N, X_T, Y_T \leq 1000$).

The second line of the input file contains a single integer number N ($1 \leq N \leq 100$) that represents the number of the shelves. Then N lines follow. Each line represents a single shelf along with its two supporting pegs, and contains five integer numbers y_i, x_i, l_i, $x1_i$, $x2_i$, separated by spaces, where:

- y_i ($0 < y_i < Y_N$) - the height of the i*th* shelf above the bottom of the niche in inches.
- x_i ($0 \leq x_i < X_N$) - the distance between the left end of the i*th* shelf and the left edge of the niche in inches.
- l_i ($0 < l_i \leq X_N - x_i$) - the length of the i*th* shelf in inches.
- $x1_i$ ($0 \leq x1_i \leq \frac{l_i}{2}$) - the distance between the left end of the i*th* shelf and its leftmost supporting peg in inches.
- $x2_i$ ($\frac{l_i}{2} \leq x2_i \leq l_i; x1_i < x2_i$) - the distance between the left end of the i*th* shelf and its rightmost supporting peg in inches.

All shelves are situated on different heights and are properly supported by their pegs. The problem is guaranteed to have a solution for the input data.

The input must be read from standard input.

Output

The output file shall contain two integer numbers separated by a space. The first one is the minimal number of pegs that are to be removed by Robinson from their original locations to place the tome. The second one is the minimal total length of planks in inches that are to be cut off during the redesign that removes the least number of pegs.

The output must be written to standard output.

Sample Input	**Sample Output**
2 11 8 3 4 4 1 1 7 1 4 4 3 7 1 6 7 2 6 3 4 2 0 3 0 3 11 8 4 6 4 1 1 7 1 4 4 3 7 1 6 7 2 6 3 4 2 0 3 0 3	0 0 1 3

J - Jumping Mario

Source file name: `jumping.c`, `jumping.cpp`, *or* `jumping.java`

Mario is in the final castle. He now needs to jump over few walls and then enter the Koopa's Chamber where he has to defeat the monster in order to save the princess. For this problem, we are only concerned with the ''jumping over the wall'' part.

You will be given the heights of N walls from left to right. Mario is currently standing on the first wall. He has to jump to the adjacent walls one after another until he reaches the last one. That means, he will make $N - 1$ jumps. A high jump is one where Mario has to jump to a taller wall, and similarly, a low jump is one where Mario has to jump to a shorter wall.

Can you find out the total number of high jumps and low jumps Mario has to make?

Input

The first line of input is an integer T ($T < 30$) that indicates the number of test cases. Each case starts with an integer N ($0 < N < 50$) that determines the number of walls. The next line gives the height of the N walls from left to right. Each height is a positive integer not exceeding 10.

The input must be read from standard input.

Output

For each case, output the case number followed by 2 integers, total high jumps and total low jumps, respectively. Look at the sample for exact format.

The output must be written to standard output.

Sample Input	Sample Output
3 8 1 4 2 2 3 5 3 4 1 9 5 1 2 3 4 5	Case 1: 4 2 Case 2: 0 0 Case 3: 4 0