CMU Fall Programming Contest: Round 1

September 24, 2011

- There are 8 problems. Do as many as you can in 4 hours.
- You *can* bring paper notes, code printouts, and books.
- You can consult these API documentation web sites:
  `http://download.oracle.com/javase/6/docs/api/`
  `http://cplusplus.com/doc`
- You *cannot* make use of any code that's in electronic form – from the internet, from the computer, or anywhere else – whether you wrote it yourself or not. You cannot use websites other than those mentioned on this page.
- The running time limit is 30 seconds, and the memory limit is 250MB. (Please note that these limits are for the server machine. The running time on your laptop might be faster or slower.)
- Input is from standard input, output is to standard output.
- Input files might consist of multiple test cases, read the input description for detials.
- The URL for the scoreboard and submitting solutions is
  `http://www.link.cs.cmu.edu/contest/current`

The Problems		
	Problem Name	Source (added after contest)
A	What are Birds?	Google Code Jam 2008 APAC-A
B	An Excel-lent problem	Greater NY 2004 B
C	Package Delivery	Topcoder SRM 363 H
D	Bacteria	Google Code Jam 2010 R2-C (Easy)
E	Power Network	SE European Regional (SEERC) 2003 G
F	Minimum Rectangle Covering	Richard Peng
G	Play Game	Topcoder SRM 217 250pt
H	High Frequency Trading	Danny Sleator

This contest was compiled and administered by Siyoung Oh, Richard Peng, Danny Sleator, and Kevin Waugh. Prizes were generously funded by IMC Financial Markets, Inc.

Carnegie Mellon

imc financial markets & asset management

A — What are Birds?

You are studying animals in a forest, and are trying to determine which animals are birds and which are not.

You do this by taking two measurements of each animal – their height and their weight. For an animal to be a bird, its height needs to be within some range, and its weight needs to be within another range, but you're not sure what the height and weight ranges are. You also know that every animal that satisfies these ranges is a bird.

You have taken some of the animals you have measured and shown them to biologists, and they have told you which are birds and which are not. This has given you some information on what the height and weight ranges for a bird must be. For the remaining animals, your program should determine if they are definitely birds, definitely not birds, or if you don't know from the information you have.

Input

The first line of the input is a number between 1 and 10, which is the number of test cases. Each test case is structured as follows:

1. One line containing an integer N ($1 \leq N \leq 1000$), the number of animals you have shown to the biologists.
2. N lines, one for each of these animals. Each line contains three space-separated positive integers, H, W, and X, where H is the height of the animal, W is the weight of the animal, and X is either the string "BIRD" or "NOT BIRD".
3. One line containing an integer M ($1 \leq M \leq 1000$), the number of animals you have not shown to the biologists.
4. M lines, one for each of these animals. Each line contains two space separated positive integters, H and W, where H is the height of the animal and W is the weight of the animal.
5. All heights and satisfy $1 \leq H, W \leq 1,000,000$.

Output

For each of the cases your program should output:

1. One line containing the string `Case #X:` where `X` is the number of the case (starting from 1) and
2. M lines, each containing one of `BIRD`, `NOT BIRD`, or `UNKNOWN`.

Example

Input	Output
3 5 1000 1000 BIRD 2000 1000 BIRD 2000 2000 BIRD 1000 2000 BIRD 1500 2010 NOT BIRD 3 1500 1500 900 900 1400 2020 3 500 700 NOT BIRD 501 700 BIRD 502 700 NOT BIRD 2 501 600 502 501 1 100 100 NOT BIRD 3 107 93 86 70 110 115	Case #1: BIRD UNKNOWN NOT BIRD Case #2: UNKNOWN NOT BIRD Case #3: UNKNOWN UNKNOWN UNKNOWN

Explanation of the Example

Case 1:

The animal "1500 1500" must be within the ranges for birds, since we know that the ranges for height and weight each include 1000 and 2000.

The animal "900 900" may or may not be a bird; we don't know if the ranges for height and weight include 900.

The animal "1400 2020" is within the height range for birds, but if 2020 was in the weight range, then the animal "1500 2010", which we know is not a bird, would also have to be within the weight range.

Case 2:

In this case we know that birds must have a height of 501. But we don't know what the weight range for a bird is, other than that it includes weight 700.

Case 3:

In this case, we know that anything with height 100 and weight 100 is not a bird, but we just don't know what birds are.

B — 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`n`C`m. n represents the row number and m represents the column number, $1 \leq n, m \leq 400,000,000$. 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.

Example

Input	Output
R1C1	A1
R3C1	A3
R1C3	C1
R299999999C26	Z299999999
R52C52	AZ52
R53C17576	YYZ53
R53C17602	YZZ53
R0C0	

C — Package Delivery

You have been assigned to deliver a number of packages from a warehouse to various destinations along a straight line. At the beginning, all the packages are at the warehouse, which is at the left end of the line. You and your truck are also at the warehouse. The packages must be delivered in increasing order of distance from the warehouse. (I.e. Deliveries to destinations near the warehouse must happen before farther ones.)

Your truck has room for `truckCapacity` packages. Driving the truck costs `fuelCost` dollars per mile, regardless of the number of packages on the truck. You may park the truck anywhere you'd like, but each time you park somewhere other than at the warehouse, it costs `parkingCost` dollars. The truck must always be parked while you deliver a package to its destination. In other words, you cannot drop packages from a moving truck. You may never leave packages anywhere other than at their destinations, at the warehouse, or in the truck.

You can also carry packages on foot, but you can only carry one package at a time. It costs `walkCost` dollars per mile to walk while carrying a package. Walking without carrying anything costs nothing.

Input

You are given a multiple test cases and each case has following format: First line contains N ($1 \leq N \leq 50$) representing the number of packages. Then, the next line contains N integers separated by white space each representing the distance (in miles) of a package's destination from the warehouse. This distance will be between 1 and 1,000,000 inclusive. On the next line, there are four integers representing `walkCost`, `fuelCost`, `parkingCost`, and `truckCapacity`. These costs will be between 0 and 1,000,000 inclusive and `truckCapacity` is positive. The last test case will end with single integer 0.

Output

For each test case, output the minimal cost to deliver all the packages. (You do not have to return the truck to the warehouse.)

Example

Input	Output
3 1 2 3 3 2 3 3 5 1 2 3 4 5 11 5 9 2 3 5 5 5 1 1 1 3 4 1 2 2 3 1000000 1 1000 2 0	13 91 6 3009

Note that in the last case there's a solution of cost 3007 if we allow a truckload to take two packages to destination 2 first, then another truckload to deliver to locations 1 and 3. But this is not allowed because the packages are not delivered in increasing order of distance.

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

1. One line containing R ($1 \leq R \leq 10$), the number of rectangles of cells that initially contain bacteria.
2. R lines containing four space-separated integers $X_1, Y_1, X_2 Y_2$. These numbers are all between 1 and 100 inclusive. Also $X_1 \leq X_2$ and $Y_1 \leq Y_2$. This indicates that all the cells with X coordinate between X_1 and X_2, inclusive, and Y coordinate between Y_1 and Y_2, 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 the bacteria all die.

Example

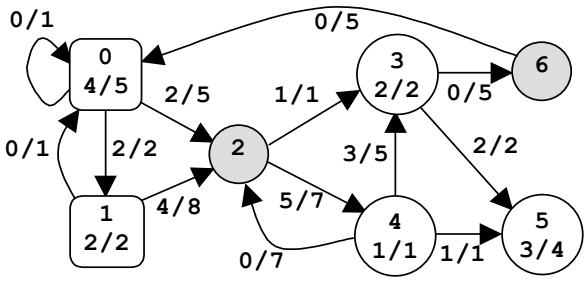
Input	Output
1 3 5 1 5 1 2 2 4 2 2 3 2 4	Case #1: 6

E — Power Network

A power network consists of nodes (power stations, consumers and dispatchers) connected by power transport lines. A node u may be supplied with an amount $s(u) \geq 0$ of power, may produce an amount $0 \leq p(u) \leq p_{\max}(u)$ of power, may consume an amount $0 \leq c(u) \leq \min(s(u), c_{\max}(u))$ of power, and may deliver an amount $d(u) = s(u)+p(u)-c(u)$ of power. The following restrictions apply: $c(u) = 0$ for any power station, $p(u) = 0$ for any consumer, and $p(u) = c(u) = 0$ for any dispatcher. There is at most one power transport line (u, v) from a node u to a node v in the net; it transports an amount $0 \leq l(u, v) \leq l_{\max}(u, v)$ of power delivered by u to v. Let $\Phi = \sum_u c(u)$ be the power consumed in the net. The problem is to compute the maximum value of it.

An example is shown in the figure below. The label x/y of power station u shows that $p(u) = x$ and $p_{\max}(u) = y$. The label x/y of consumer u shows that $c(u) = x$ and $c_{\max}(u) = y$. The label x/y of power transport line (u, v) shows that $l(u, v) = x$ and $l_{\max}(u, v) = y$. The power consumed is $\Phi = 6$. Notice that there are other possible states of the network but the value of Φ cannot exceed 6.

u	type	s(u)	p(u)	c(u)	d(u)
0	power station	0	4	0	4
1		2	2	0	4
3	consumer	4	0	2	2
4		5	0	1	4
5		3	0	3	0
2	dispatcher	6	0	0	6
6		0	0	0	0



Input

There are several data sets in the input text file. Each data set encodes a power network. It starts with four integers: $0 \leq n \leq 100$ (nodes), $0 \leq n_p \leq n$ (power stations), $0 \leq n_c \leq n$ (consumers), and $0 \leq m \leq n^2$ (power transport lines). This is followed by m data triplets of the form $(u, v)z$, where u and v are node identifiers (starting from 0) and $0 \leq z \leq 1000$ is the value of $l_{\max}(u, v)$.

This is followed by n_p doublets $(u)z$, where u is the identifier of a power station and $0 \leq z \leq 10,000$ is the value of $p_{\max}(u)$. The data set ends with n_c doublets $(u)z$, where u is the identifier of a consumer and $0 \leq z \leq 10,000$ is the value of $c_{\max}(u)$. All input numbers are integers, except the $(u, v)z$ triplets and the $(u)z$ doublets, which do not contain white spaces. (Other than this, white spaces and newlines can occur freely in input.) The input is terminated by end-of-file.

Output

For each data set from the input, the program prints on the standard output the maximum amount of power that can be consumed in the corresponding network. Each result has an integral value and is printed from on a separate line.

Example

Input	Output
2 1 1 2 (0,1)20 (1,0)10 (0)15 (1)20 7 2 3 13 (0,0)1 (0,1)2 (0,2)5 (1,0)1 (1,2)8 (2,3)1 (2,4)7 (3,5)2 (3,6)5 (4,2)7 (4,3)5 (4,5)1 (6,0)5 (0)5 (1)2 (3)2 (4)1 (5)4	15 6

The sample input contains two data sets. The first data set encodes a network with 2 nodes, power station 0 with $p_{\max}(0) = 15$ and consumer 1 with $c_{\max}(1) = 20$, and 2 power transport lines with $l_{\max}(0, 1) = 20$ and $l_{\max}(1, 0) = 10$. The maximum value of Φ is 15. The second data set encodes the network from the figure above.

F — 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 ($-1000 \leq x_i, y_i \leq 1000$).

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. Answers within an absolute or relative error of 0.001 will be accepted as correct.

Example

Input	Output
2 2 0 0 1 1 5 0 0 0 2 2 0 2 2 1 1	0 4

G — Play Game

You are playing a computer game and a big fight is planned between two armies of the same size. 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. 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 the computer) have. The next line contains N space-separated integers representing the power of your units. The last line also contains N integers representing the arrangement of the computer's units (in the above format). Each unit has power in the range 1 to 1,000 inclusive. The input is terminated by a line containing "0".

Output

On the first line, you should print out the maximum total strength of your units that are not captured. Then, on the following line, you should print out the arrangement of your units. If there are multiple possible arrangements with the same maximum total, print any one of them. Follow the format below.

Example

Input	Output
5 5 15 100 1 5 5 15 100 1 5 0	120 100 1 5 5 15

H — High Frequency Trading

You have the stock price every millisecond over a period of time. At the beginning of each millisecond you own a certain amount of stock and have a certain amount of cash on hand. You can buy some more stock with the cash you have on hand, or you can sell some of your stock for cash. You can even do both. Write a program that takes (1) the price at every millisecond, (2) an initial amount of cash, and, (3) a limit on the total number of trades you can make. The program should output the log of the maximum amount of cash you can have at the end. (The stock you have at the end, if any, is not counted.)

Input

The input consists of one or more problem instances, followed by a line containing "0". The instance begins with a line containing n, the number of time periods. This is followed by a line containing c, your initial amount of cash in dollars (a positive integer, at most 1000), and t, the maximum number of trades that you're allowed. This is followed by n lines, each of which is the price of a share of stock (a positive integer that is at most 1000) at the start of each millisecond. $1 \leq n \leq 5000$, $0 \leq t \leq 1000$.

Output

For each instance, output the the log (base 10) of the maximum amount of money that you can have at the end. The input is engineered such that the output of a correct program will be less than 100. An absolute error of .0001 is acceptable.

Example

Input	Output
4 1 2 4 2 3 5 8 500 6 1 2 5 2 3 4 1 2 0	0.39794 4.00000