



ACM ICPC Asia Regional 2011 Kuala Lumpur

Problem Set

This problem set contains 9 questions (A-I)

Input: Standard Input
Output: Standard Output

ACM ICPC Malaysia Office
Kulliyah of Information and Communication Technology
International Islamic University Malaysia



acm International Collegiate Programming Contest



event sponsor

A

Smooth Visualization

Input: Standard Input
Output: Standard Output



One possible method to visualize a number with many digits whose values are from 1 to 7, inclusive, is to visualize each digit by a vertical column of '+' on a background canvas of '*'. The digit seven (7) would be visualized by a vertical column of seven '+', the digit six (6) would be visualized by a vertical column of six '+' and so on. For example, the number "1425" will be visualized as

```
***+
**++
**++
**++
*+++
++++
```

Such representation is seen as jagged, as the difference between one column of '+' and the next is greater than 1, and may be considered not aesthetically pleasing by some. You are asked to add the minimum number of columns to eliminate its jagged nature and construct the following smoother visualization

```
*****+
***+***++
*++++*+++
*+++++++
+++++++
```

where the height of two adjacent columns of '+' differ by no more than one.

Input

The input starts with an integer N ($1 \leq N \leq 100$), on a line by itself, that indicates the number of inputs to be visualized. Each of the following N lines contains a single number which is described by K digits whose values are in the range of 1 to 7, inclusive. A number does not contain any blank spaces and the value of its size K is in the range of 1 to 100, inclusive.

Output

For each input, the output consists of a smoothed visualization as explained above and demonstrated in the samples below.

Sample Input	Output for the Sample Input
3 1425 114242 1726	<pre> *****+ ***+***++ **+++*+++ *+++++++ +++++++ +++++++ ***+***+** **+++*+++* **+++++++ +++++++ *****+***** *****+*****+ ***+++++*****+ **+++++++**+++ **+++++++*+++ *+++++++*++++ *+++++++ +++++++ </pre>



acm International Collegiate
Programming Contest

IBM | event
sponsor

B

Arnook's Defensive Line

Input: Standard Input
Output: Standard Output



Based on the latest intelligence reports, Chief Arnook of the northern tribe has become suspicious of the warrior nations that dwell south of the border. The chief has ordered his warriors to protect the southern border which runs parallel to the 54° latitude line and stretches between the longitudes of 1° to $1000,000,000^\circ$, inclusive.

Each warrior is assigned the task of protecting a segment of the border defined to lie between longitudes “a” and “b”, inclusive. No two warriors are assigned to protect the exact same segment. Bound by loyalty to his chief, a warrior will inform the chief upon his arrival at his appointed post and will never leave once he arrives.

Your task is to write a program that performs the following two operations

- + a b a warrior assumes his position and protects the segment between longitudes “a” and “b”, inclusive.
- ? c d computes the number of warriors who completely protect the segment between longitudes “c” and “d”, inclusive. The segment between the longitudes “c” and “d”, inclusive, is said to be completely protected by a warrior X if and only if warrior X protects a segment between “a” and “b”, inclusive, and $a \leq c < d \leq b$.

to help Chief Arnook track the status of his border protection.

Input

The input starts with an integer N ($1 \leq N \leq 500000$), on a line by itself, that indicates the number of operations. Each of the following N lines contains one operation. The description of an operation consists of a character “+” or “?” followed by two integers on a line by itself. The entries on a line are separated by single blank spaces.

Output

There is one output line for each input line that starts with the operation “?”. The output consists of a single integer that represents the number of warriors who completely protect the corresponding segment at the time.

There is no output for input lines that start with the character “+”.

Sample Input	Output for the Sample Input
9	2
+ 5 10	3
+ 7 20	4
+ 3 15	3
? 9 12	
+ 10 20	
? 8 9	
+ 6 30	
? 8 9	
? 9 12	



C

Equivalence

Input: Standard Input
Output: Standard Output



In preparation for their advanced mathematics paper, Alice and Bob are attempting to solve problems whose answers are supposed to be complicated formulas. Their answers may appear to be different even though they are both correct.

Your task is to write a program to read different formulas and determine whether or not they are arithmetically equivalent.

Input

The input starts with an integer T ($1 \leq T \leq 20$), on a line by itself, that indicates the number of cases. Each case consists of two arithmetic expressions, each one a separate line with at most 80 characters. An expression contains one or more of the following:

- Single letter variables (case insensitive).
- Single digit numbers.
- Matched left and right parentheses.
- Binary operators $+$, $-$ and $*$ which are used for addition, subtraction and multiplication, respectively.
- Arbitrary number of blank or tab characters between above tokens.

Expressions are syntactically correct and evaluated from left to right with equal precedence (priority) for all operators. The coefficients and exponents of the variables are integers.

Output

For each case the output consists of one line that contains “YES” if the two input formulas are arithmetically equivalent, otherwise it contains “NO”.

Sample Input	Output for the Sample Input
2	YES
$a*2-(a+c)+((a+c+e)*2)$	NO
$3*a+c+(2*e)$	
$(a-b)*(a-b)$	
$(a*a)-(2*a*b)-(b*b)$	



acm International Collegiate
Programming Contest

IBM | event
sponsor

D

Tree Inspections

Input: Standard Input
Output: Standard Output



Fruit fly, like *Drosophila suzukii*, is a fruit crop pest and is a serious economic threat to soft summer fruits. The fertilized female fruit fly searches for ripe fruit, inserts its serrated ovipositor to pierce the skin and deposits a clutch of eggs per insertion. The larvae hatch and grow in the fruit and destroy the fruit's commercial value. Regular inspection of fruit farms are required to detect infected trees and destroy them as a way of stopping the spread of infestation to other farms.

In our region, fruit trees are planted in rectangular shaped farms with planned walking paths for fruit collectors and tree inspectors. The location of each tree is described by its x- and y-coordinates. Each walking path is either a horizontal line or a vertical line that stretches from one side of the farm to the other side. Horizontal and vertical paths are described by their y- and x-coordinates, respectively.

It is required, by law, that at least 60% of the trees in a farm be visible, where a tree is visible only if it can be viewed by an inspector standing on some path and facing perpendicular to the path. There must be no intervening trees that obstruct the inspector's view.

Your task is to write a program to test if a fruit farm passes the 60% visibility requirement.

Input

The input starts with an integer F ($1 < F \leq 10$), on a line by itself, that indicates the number of farms to be processed. The description for each farm starts with a line that contains two integers T , and P . T represents the number of trees and P represents the number of paths. Each of the following T lines contains two integers that represent the x-coordinate followed by the y-coordinate of one tree. Each of the following P lines starts with the letter H (indicating a horizontal path) or the letter V (indicating a vertical path) followed by an integer that represents the corresponding coordinate.

$1 \leq (T, P) \leq 100000$, and all coordinates lie between -1000000 and 1000000 , inclusive.

Output

For each farm, the output consists of a single line that contains either

1. PASSED (if the farm satisfies the 60% rule), or
2. FAILED

Sample Input	Output for the Sample Input
1 6 3 -1 3 4 2 6 2 6 3 6 4 4 3 H -1 V 0 H 5	PASSED



acm International Collegiate Programming Contest



event sponsor

E

Social Holidaying

Input: Standard Input
Output: Standard Output



“The Holiday Company” owns a large number of bungalows that were built in the olden days to host large, or extended, families during the holiday season. Instead of borrowing large sums of money to renovate the bungalows to smaller sizes suitable for today's families, the company's manager introduced the new concept of “social holidaying”.

The idea of “social holidaying” is to accommodate two and only two families in a bungalow if the sum of sizes of the two families exactly equals the number of available beds in the bungalow.

For example, two is the largest number of possible pairing for a set of holiday requests by families with sizes 1, 2, 3, 5, and 5 in a resort with bungalows of sizes 3, 6, and 10. The possible pairings are (1 with 2) and (5 with 5).

Your task is to write a program to calculate the largest number of possible pairings of holiday requests. The input to your program consists of a set **R** of n holiday requests and a set **B** of m bungalow sizes. You should assume that there are a sufficiently large number of bungalows available for each size in **B**.

Input

The input starts with an integer P ($1 \leq P \leq 100$), on a line by itself, that indicates the number of problem descriptions. Each problem description consists of three lines: The 1st line contains two integers, n and m that represent the sizes of the sets \mathbf{R} and \mathbf{B} , respectively. The 2nd line contains the n family sizes in \mathbf{R} , and the 3rd line contains the m sizes of bungalows in \mathbf{B} . The values in each line are separated by single spaces. $2 \leq n \leq 400$, and $1 \leq m \leq 100$.

Output

For each problem, the output is a single line consisting of an integer that is the maximum number of possible pairings.

Sample Input	Output for the Sample Input
2	3
6 4	2
1 2 3 4 4 5	
6 9 3 5	
5 4	
1 2 3 4 5	
6 9 3 5	



acm International Collegiate
Programming Contest



event
sponsor

F

Orienteering

Input: Standard Input
Output: Standard Output



Orienteering involves running through unfamiliar territory, using a map and a compass to navigate to various control points marked on the map. In the most common form of the sport, the order in which the control sites must be visited is set in advance by the race organizers, and the winner of a race is the runner who visits all the controls in the prescribed order and arrives at the finish line in the shortest amount of time. Thus the goal is to visit all control points (in order) as fast as possible.

An obscure variant of the sport is *Half-Score Orienteering*, in which the organizers define the order in which the control points must be visited, choose a time limit and assign a score to each control point, generally correlated to its difficulty and distance from the starting point of the race. Each runner begins this kind of race by estimating how far he can run in the time available and then chooses a subset of the control points. The runner's aim is for a high-scoring subsequence that forms a route following the general outline of the organizers' course (and ordering) but skips undesired control points. The total length of the route is close to, but not exceeding, his distance estimate.

The start and finish points for each race are at the origin $(0, 0)$. Assume that the runners navigate perfectly along straight-line paths between control points (and to/from the start and finish). Therefore a viable route for a runner starts at the origin, passes through a subsequence of the control points, in the order of appearance in the input, and then returns to the origin. The total length must not exceed his distance estimate.

After a race, the runners are always curious as to whether they could have done better by aiming for a different subsequence of the available control points. Your task is to write a program to determine the maximum score available for each runner in a race.

Input

Input contains a number of races. The description of each race consists of:

The first line contains an integer, n , that represents the number of control points in the race ($1 \leq n \leq 30$).

Each of the following n lines contains three integers, separated by spaces, that represents data for one control point. The three integers are x , y , and s , where (x, y) are the coordinates of the control point in meters ($-5000 \leq x, y \leq 5000$) and s is the control point's score ($10 \leq s \leq 200$).

A number of lines, with one line for each runner in the race, follows. "# 0" on a line by itself terminates the runners data. Each of these lines contains the runner's name and an integer d , separated by a single space. The name is a single word of up to 60 characters with no blank spaces, and d is the distance in meters that the runner can travel in the time available ($0 \leq d \leq 10000$).

The last line of input contains a "0". This line should not be processed.

Output

Output for each race starts with a line 'Race r ' stating the race number, starting with 1. This will be followed by one line for each of the runners in the race, in the order in which they appear in the race input, containing the runner's name and the score of the highest scoring viable route available to that runner, formatted as 'name: score'.

Sample Input	Output for the Sample Input
5 750 -800 30 1500 0 50 750 750 60 -1250 750 70 -1000 -500 50 Chris 7000 Karl 6500 Tania 5000 # 0 4 500 0 10 0 500 10 -500 0 10 0 -500 10 Hanny 2100 Lizzie 1800 # 0 0	Race 1 Chris: 230 Karl: 180 Tania: 140 Race 2 Hanny: 20 Lizzie: 20



acm International Collegiate Programming Contest



event sponsor

G

Writings on the Wall

Input: Standard Input
Output: Standard Output



After years of studying the drawings on a wall, in an ancient Malaysian temple, a famous archaeologist believes that she has unlocked its secret. Her theory is that the drawings are characters of an ancient language and that they are organized into long strings, which she identified to her student. The student took pictures of the drawings, for analysis back in the office, and needed two photos to record each string: one which includes the left most part of the string and one with the right most part of the string .

Back at the office they realized that there are only 26 different characters, which they mapped into the English alphabet 'a' to 'z'. Unfortunately, the student forgot to put markers on the wall before he took the pictures and there may be some overlapping in the photos. For example, the photo of the left part “xyzabcabc” can be combined with the photo of the right part “abcabcxyz” in the following three different ways:

```
xyzabcabcabcxyz  
xyzabcabcabcxyz  
xyzabcabcxyz
```

Your task is analyze the available pairs of left and right photos and compute the number of possible lengths of their original strings.

Input

The input starts with an integer T ($1 \leq T \leq 100$), on a line by itself, that indicates the number of cases. Each case contains two strings, separated by a single blank space on a line by themselves, which describe the left and right photos respectively. Each string consists of lowercase alphabet ('a', 'b', ..., 'z'), does not contain blank spaces and has a length between 1 and 50000, inclusive.

Output

For each case, the output consists of a single integer on a line by itself. The integer represents the number of possible string lengths.

Sample Input	Output for the Sample Input
3	3
xyzabcabc abcabcxyz	2
xyzabcd abcdxyz	1
acmicpc kualalumpur	



acm International Collegiate Programming Contest



event sponsor

H

Robotic *Traceur*

Input: Standard Input
Output: Standard Output



Parkour is a method of movement focused on moving around obstacles with speed and efficiency. Originally developed in France, the main purpose of the discipline is to teach participants how to move through their environment by vaulting, rolling, running, climbing and jumping. After accessing a demo of *Parkour* on YouTube, *Robo* (a second generation intelligent robot model) made up its mind to practice a simple version of *Parkour* in its workspace and to become the first robotic *Traceur* (*parkour* practitioner).

Robo prepares a *parkour* practice by placing dots on the floor of its workshop with two dots identified as the starting and finishing locations. *Robo* begins the practice by standing on the starting location with one of its two legs, and then reaches with its other leg for another dot and shifts its weight onto it. The leg mechanisms allow *Robo* to reach a dot at a distance that equals the sum of the lengths of its two legs. The process is repeated until the finishing location is reached, if possible.

Your task is to write a program for *Robo* that analyzes a practice setup and computes the minimum number of moves required to reach the finishing location, if possible.

Input

The input starts with an integer T ($1 \leq T \leq 100$), on a line by itself, that indicates the number of practices. The description of each practice starts with a line that contains three integers N , S and F followed by two decimals $L1$ and $L2$. N represents the number of dots, S represents the index of the starting dot, and F represents the index of the finishing dot. $L1$ and $L2$ are positive decimal values, with at most three digit precision, which represent the lengths of the *Robo's* two legs.

$$2 \leq N \leq 1000, 1 \leq (S, F) \leq N, \text{ and } 0 < (L1, L2) \leq 30000.$$

Each of the following N lines contains two integers that represent the x - and y -coordinates of a dot. The j th line contains the coordinates of the j th dot, $1 \leq j \leq N$. The locations of the dots are distinct and their coordinates values are between -30000 and 30000 , inclusive.

Output

For each practice, the output consists of a single line that contains the minimum number of moves required by *Robo* to reach the finishing location. If this is not possible, print "Impossible".

Sample Input	Output for the Sample Input
2	Impossible
4 1 4 2.000 1.000	8
0 0	
0 4	
4 0	
4 4	
9 1 4 2.000 3.000	
0 7	
-6 2	
-3 3	
6 2	
-6 -3	
3 -3	
6 -3	
-3 -7	
0 -7	



acm International Collegiate Programming Contest



event sponsor

I

Shortest Leash

Input: Standard Input
Output: Standard Output



John has a dog that he calls Euclid. During their daily walk in the park Euclid only runs a predictable distance parallel to each of a number of n distinct lines, once and only once, in a predictable order before returning back to him. Therefore, each segment of Euclid's run is a vector that is described by a pair of x and y displacements. Euclid may run in the direction of the vector or in the opposite direction (that is, displacement of $-x$ and $-y$).

John does not like to be dragged around and he would like to know the length of the shortest leash he has to buy for his dog. Your task is to help John by writing a program to compute the maximum distance John's dog can be located away from him.

Input

The input contains a number of instances. Each instance starts with a line containing the integer n , $0 \leq n \leq 100$. Each of the next n lines contains a pair of integers x_i and y_i that represents the i th vector.

The value of n equals zero indicates the end of input and should not be processed. The absolute values of all the coordinates are less than 1000.

Output

For each test case, output consists of a single line that contains the maximum distance the dog can be away from John. The distance must be rounded and displayed as three decimal places.

Reminder: Rounding a positive number R.xxyy to three decimal places

- If the fourth decimal place is less than 5, then the rounded value is R.xxx
- Otherwise, the rounded value is R.xxx + 0.001

Examples are: for the value of 10.3463 the output should be 10.346, and for the value of 10.3695 the output is either 10.37 or 10.370

Sample Input	Output for the Sample Input
3	3.000
1 1	5.099
0 1	37.336
-1 1	
2	
4 0	
1 1	
7	
1 3	
-2 -7	
7 8	
-2 9	
-7 -3	
4 -3	
-2 -2	
0	