The 2010 ACM ICPC World Finals Warmup 2

Sponsored by IBM

Hosted by
Valladolid Online Judge

23rd January, 2010
You get 13 Pages
10 Problems
&
300 Minutes

A

Super Heronian Triangle

Input: Standard Input
Output: Standard Output

A Heronian Triangle has sides with integer length and integer area. For this problem, a Heronian Triangle whose radius of circum-circle is also integer is called Super Heronian Triangle (SHT). Given the radius of Circum Circle and area of an SHT, you will have to find such a triangle all whose side lengths are less than 40001.

Input

The input file contains at most 300 lines of inputs. Each line contains two integers R ($0 < R < 500000$) and A ($0 < A < 700000000$). Here R and A denotes the radius of circum circle and Area of an unknown SHT. Input is terminated by a line containing two zeroes which obviously should not be processed.

Output

For each line input produce one or more line of output. If there is no SHT with the given radius and area and all side with lengths less than 40001, print three -1 (Minus 1) separated by single space. Other wise print the length of sides of the SHT in ascending order. If there is more than one such triangle print the one whose smallest side has minimum length.

Sample Input	Output for Sample Input

5 24
5 23
0 0
	6 8 10
	-1 -1 -1

Problemsetter: Shahriar Manzoor
Special Thanks: Samee Zahur

B

Race to 1

Input: Standard Input
Output: Standard Output

Dilu have learned a new thing about integers, which is - any positive integer greater than 1 can be divided by at least one prime number less than or equal to that number. So, he is now playing with this property. He selects a number **N**. And he calls this **D**.

In each turn he randomly chooses a prime number less than or equal to **D**. If **D** is divisible by the prime number then he divides **D** by the prime number to obtain new **D**. Otherwise he keeps the old **D**. He repeats this procedure until **D** becomes 1. What is the expected number of moves required for **N** to become 1.

[We say that an integer is said to be prime if its divisible by exactly two different integers. So, 1 is not a prime, by definition. List of first few primes are 2, 3, 5, 7, 11, …]

Input

Input will start with an integer **T (T <= 1000)**, which indicates the number of test cases. Each of the next **T** lines will contain one integer **N (1 <= N <= 1000000)**.

Output

For each test case output a single line giving the case number followed by the expected number of turn required. Errors up to 1e-6 will be accepted.

Sample Input

```
3
1
3
13
```

Output for Sample Input

```
Case 1: 0.0000000000
Case 2: 2.0000000000
Case 3: 6.0000000000
```

Problemsetter: Md. Arifuzzaman Arif
Special Thanks: Sohel Hafiz

C

Hats Off

Input: Standard Input
Output: Standard Output

It's the Wild West. But things are not quite well here, now. Last few months, many trains have been robbed by the mysterious and vicious bandit called 'El Diablo'. The railroad company has offered a reward of about **$150000** for the one who can stop him. That's why John Cooper is trying to catch him.

But the task is not that easy as it looks. That's why John made a team and planned to chase after the 'El Diablo'. After a lot of hard work and with mild plans, they finally managed to surround him in a room. But the problem is that the only door of the room is locked and it can't be unlocked other than finding a combination.

There are **n** stones scattered outside the room. Each of them is equal in size but labeled by an integer number. The door has **n** holes numbered from **1** to **n**. Any hole can contain at most one stone. In the upper side of the door, **m** pairs of integers are written. Each of them denotes a range of some consecutive holes.

John is trying to arrange the stones in the holes such that, the maximum difference of the maximum labeled stone and the minimum labeled stone in any range is minimized. If John can place them in the best possible way, then the door will be unlocked.

El Diablo will escape if it takes more than **5** hours. Cause he is digging a hole which leads to a cave. If he reaches the cave, all the plans and hark works will be in vain, because he can escape through the cave. That's why John is seeking your help. Since John is not a programmer, he asks you to write a program which will solve this problem and unlock the door. So, Hats off to you for helping John. May be you will get a share of the reward if you can unlock the door.

Input

First line of the input will contain the number of cases **T (<=300)**. Then **T** cases follow.

Each case will start with an integer **n (1 <= n <= 30)**. The next line will contain **n** integers denoting the labels of the stones. The next line will contain an integer **m (1 <= m <= 5)**. Each of the next **m** lines will contain two integers **a b (1 <= a <= b <= n)** denoting all the holes from **a** to **b** (inclusive). Each of the integers will be fit into a **32** bit signed integer.

Output

For each case, print the case number and the desired result as shown below.

Sample Input

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

Output for Sample Input

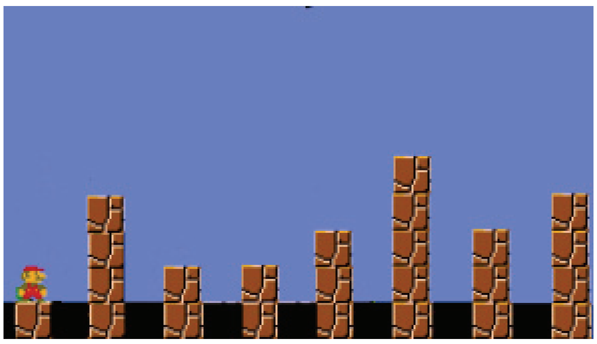
```
Case 1: 2
Case 2: 3
```

Problemsetter: Jane Alam Jan
Special Thanks: Manzurur Rahman Khan

D

Jumping Mario

Input: Standard Input
Output: Standard Output



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.

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.

Sample Input

```
3
8
1 4 2 2 3 5 3 4
1
9
5
1 2 3 4 5
```

Output for Sample Input

```
Case 1: 4 2
Case 2: 0 0
Case 3: 4 0
```

Problemsetter: Sohel Hafiz
Special Thanks: Shamim Hafiz

E

Component Placement

Input: Standard Input
Output: Standard Output

In circuit design, component placement is an important step in the design automation. Inferior placements may affect the performance and manufacturing cost.

You are given a PCB (Printed Circuit Board). It's a large green board. You can place components on either side of the PCB. However, cost of placing a component on both sides are not the same. You are given **N** components. For each component $\mathbf{c_i}$, cost of placing it on the top layer is $\mathbf{X_i}$ and on the bottom layer is $\mathbf{Y_i}$.

These components may interact with each other. If both the components are on the same side of the PCB, the interconnection cost is negligible. But, if they are on different sides, their interconnection is costly. For each such interconnection **j**, the cost will be $\mathbf{Z_j}$.

Finally, some design issues restricts some components to be on the top side or bottom side. Now, find the minimum cost to place the components.

Input

First line contains a positive integer **T (T<= 50)** that denotes the number of test cases.

Each test case starts with 2 integers **N (1 <= N <= 200)** and **M (0 <= M <= 100000, M <= N * (N-1) / 2)**, the number of components and number of interconnections. This will be followed by ***N*** integers in a line, each between 1 and 10000000 (inclusive), where **i**-th of it describes the cost of placing the component on the top layer. The next line contains **N** more integers, each between 1 and 10000000 (inclusive), where **i**-th of it denotes the cost of placing it on the bottom layer. The next line contains **N** more integers, each will be either 0, -1 or +1, where

- -1 means **i**-th component can only be placed on the bottom
- +1 means **i**-th component can only be placed on the top
- 0 means the component can be placed on either side

Then there will be **M** lines, each containing three integers, **p**, **q**, and **r (1 <= p, q <= N, 1 <= r <= 10000000)**, denoting that, **p** and **q-**th component has to be interconnected and if they are on different layers, the cost of interconnection will be **r*.*** There will be at most one interconnection between any pair or components.

Output

For each test case, output the minimum cost to place the components.

Sample Input

```
5
4 0
5 6 7 8
8 7 6 5
0 0 0 0
4 2
5 6 7 8
8 7 6 5
0 0 0 0
1 3 10
2 4 10
4 3
5 6 7 8
8 7 6 5
0 0 0 0
1 3 10
2 4 10
2 3 1
4 3
5 6 7 8
30 31 32 33
0 0 0 0
1 3 10
2 4 10
2 3 1
4 3
5 6 7 8
8 7 6 5
-1 0 0 1
1 2 10
3 4 10
2 3 1
```

Output for Sample Input

```
22
24
25
26
31
```

Problemsetter: Manzurur Rahman Khan
Special Thanks: Samee Zahur

F

Racing Car Computer

Input: Standard Input
Output: Standard Output

The racing cars of today are equipped with so many sophisticated equipment. Introduction of a new visual transducer which is interfaced with the on-board computer can tell you on the fly how many cars are ahead of you while how many are trailing. There are **N** cars on a racing track. Each has an on-board computer with the new feature. During the race, every single car's computer keeps displaying two integers, **a** (The number of cars in front of him) & **b** (The number of cars behind him) for a particular moment. It is possible that at some time, some of the cars are racing side by side i.e. they are exactly at the same location. A car will not consider any other car at the same location to be a leading or trailing car.

Now, it is suspected that some of the new transducers are not working properly inside such high speed vehicles. The report with all computers' data generated at a particular timestamp is reported to you. You are to determine the minimum number of cars that have faulty data.

Input

Each test case begins with an integer **N (1<=N<=1000),** the number of cars on a track . The next N lines each has two integers - **a** & **b (0<=a,b<=1500)** for a particular car.

The last test case is followed by a line with a single 0 indicating the end of input.

Output

For each test case, print a line in the format, "**Case X: Y**", where **X** is the case number & **Y** is the minimum number of cars that must have faulty data according to the report.

Sample Input

```
4
2 2
0 0
0 2
3 1
1
1 1
0
```

Output for Sample Input

```
Case 1: 3
Case 2: 1
```

Problemsetter: Mohammad Mahmudur Rahman
Special Thanks: Arifuzzman Arif

G World of Knots

Input: Standard Input
Output: Standard Output

Today, we will be unraveling the mysteries of one of the most twisted and curly beasts in all existence – knots! The problem here is simple: given the picture of a string laid out curled on a table, you are to determine if it will curl up into a knot when the ends are pulled out.

```
...............
...+------+....
...|......|....
...|...+--H----
...|...|..|....
---I---H--+....
...|...|.......
...+---+.......
...............
```

The input will consist of a diagram of one single continuous string described in a two dimensional grid of characters such as shown above. The '-' and '|' characters represent a horizontal and vertical segment of the string, respectively. '+' characters represent corners where the string turns at right angles on the table. 'I' or 'H' characters represent locations where parts of the strings cross:

- 'H' represents locations where the horizontal string passes *over* the vertical string
- 'I' represents locations where the horizontal string passes *under* the vertical string

The dot character, '.', obviously, represents empty spaces of the table unoccupied by the string. Two horizontally adjacent nonempty spaces of the table are connected by the string iff neither of them are '|' characters. Similarly, vertically adjacent nonempty spaces are connected by the string iff neither of them are '-' characters. Inputs will be such that every '+' character will represent a proper corner where the string turns at a unambiguously at a right angle: formally, every '+' character will be connected to exactly one vertically adjacent and exactly one horizontally adjacent space.

Moreover, to further simplify matters, you can assume that the only characters along the border of the given diagram, other than dots, will represent endpoints of the string. In short, the input will unambiguously specify a valid piece of string starting and ending at the border of the input diagram. Finally, assume that the string has negligible width and is made of a very smooth material, so that parts of the string can easily slide over each other with negligible friction.

Input

The input will consist of at most 80 test cases. Each test case will start with a single line containing two integers, *r* and *c*, indicating that the size of the diagram for that case is of *r* rows and *c* columns. This line will be followed by *r* more lines, each with exactly *c* characters, with characters in each line representing a row of the diagram. You may assume that $2 \le r,c \le 120$. Input is terminated by a line containing two zeroes. This line should not be processed.

Output

The output should consist of exactly one line for each test case in the format
"Case *c*: *Response*",

where *c* is the test case serial number, starting from 1, and *Response* is either the string "straightened" or "knotted" (without the quotes) depending on whether the string will straighten out or coil up into a knot, respectively.

See the sample for further clarifications.

Sample Input

```
9 15
...............
...+------+....
...|......|....
...|...+--H----
...|...|..|....
---I---H--+....
...|...|.......
...+---+.......
...............
9 15
...............
...+------+....
...|......|....
...|...+--I----
...|...|..|....
---I---H--+....
...|...|.......
...+---+.......
...............
0 0
```

Output for Sample Input

```
Case 1: knotted
Case 2: straightened
```

Problemsetter: Samee Zahur
Special Thanks: Manzurur Rahman Khan

H

Lattice Point or Not

Input: Standard Input
Output: Standard Output

Now a days a very common problem is: "The coordinate of two points in Cartesian coordinate system is (200, 300) and (4000, 5000). If these two points are connected we get a line segment. How many lattice points are there on this line segment." You will have to do a similar task in this problem – the only difference is that the terminal coordinates can be fractions.

Input

First line of the input file contains a positive integer N (N<=50000) that denotes how many lines of inputs follow. This line is followed by N lines each of which contains four floating-point numbers x1, y1, x2, y2 ($0< |x_1|, |y_1|, |x_2|, |y_2| <=200000$). These floating-point numbers has exactly one digit after the decimal point.

Output

For each line of input except the first line produce one line of output. This line contains an integer which denotes how many lattice points are there on the line segment that connects the two points (x1, y1) and (x2, y2).

Sample Input	Output for Sample Input

3
10.1 10.1 11.2 11.2
10.2 100.3 300.3 11.1
1.0 1.0 2.0 2.0
	1
	0
	2

Problemsetter: Shahriar Manzoor
Special Thanks: Derek Kisman

I

All Souls Night

Input: Standard Input
Output: Standard Output

"Standing on the bridge that crosses the river that goes out to the sea
The wind is full of a thousand voices, they pass by the bridge and me
I can see the lights in the distance trembling in the dark cloak of night
Candles and lanterns are dancing, dancing a waltz on all souls night."

In the traditional all souls night celebration, souls of the departed are commemorated by sending candle-lit lanterns on out waterways leading to the oceans, sometimes in little boats. A single lantern represents an individual soul drifting away to the next world. Let's assume the position of a lantern just at the moment of entering the next world is given by the coordinates of a single point in 3-d space. A pair of souls can communicate with each other by straight-lined light rays joining themselves.

As soon as the souls enter afterlife, they are covered by a mysterious veil which separates themselves from the distant living world. The veil should be placed in such a way that the communication beams between any two souls are entirely contained within the veil in order to prevent the living world people overhearing them. Now, as the weavers of heaven are tired of going on knitting veils for an eternity, they ask for your help. You are asked to calculate the minimum surface area of the veil required to cover a group of souls fulfilling the restrictions given above. Help Heaven!

Input

The input file will contain multiple test cases. Each case begins with an integer **N (3 < N < 26)**, the number of souls in a group. Each of the following **N** lines contains 3 integers giving the **x**, **y** and **z** coordinates of a single soul (**-100 <= x, y, z <= 100)**. They will be positioned in such a way that there won't be more than 3 souls in any polygonal face of the veil. The end of input will be marked by a case with **N = 0**. This case should not be processed.

Output

For each test case, print a line in the format, "**Case X: Y**", where **X** is the case number & **Y** is the minimum required surface area of the veil. Print 2 digits after the decimal point for **Y**.

Sample Input	Output for Sample Input

4
0 0 0
4 0 0
0 4 0
0 0 4
0
	Case 1: 37.86

Problemsetter: Mohammad Mahmudur Rahman
Special Thanks: Jane Alam Jan

J

Lighting Away

Input: Standard Input
Output: Standard Output

Ronju is a night-guard at the "Lavish office buildings Ltd." headquarters. The office has a large grass field in front of the building. So every day, when Ronju comes to duty at the evening, it is his duty to turn on all the lights in the field. However, given the large size of the field and the large number of lights, it is very tiring for him to walk to each and every individual light to turn it on.

So he has devised an ingenious plan – he will swap the switches for light-sensitive triggers. A local electronic store nearby sells these funny trigger switches at a very cheap price. Once installed at a light-post, it will automatically turn that light on whenever it can sense some other light lighting up nearby. So from now on, Ronju can just manually flip a few switches, and the light from those will trigger nearby sensors, which will in turn light up some more lights nearby, and so on, gradually lighting up the whole field.

Now Ronju wonders: how many switches does he have to flip manually for this?

Input

The input starts with an integer **T**, the number of test cases to follow. Each test case will start with two integers **N** (**1 <= N <= 10000**) and **M** (**0 <= M <= 100000**), where **N** is the number of lights in the field, and **M** more lines of input follows in this input case. Each of these extra **M** lines will have two integers **a** and **b** separated by a space, where **1 <= a, b <= N**, indicating that if the light **a** lights up, it will trigger the light **b** to turn on as well (according to their distance, brightness, sensor sensitivity, orientation and other complicated factors). Finally, every test case in the input will be followed by a blank line.

Output

For each input test case, the output must be a single line of the format "**Case k: c**" where **k** is the case number starting with 1, and **c** is the minimum number of lights that Ronju must turn on manually before all the lights in the whole field gets lit up.

Sample Input

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

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

Output for Sample Input

```
Case 1: 2
Case 2: 2
```

Problemsetter: Samee Zahur
Special Thanks: Jane Alam Jan