ACM ICPC World Finals 2014 Warmup

Hosted at Codemarshal

14th June 2014
You get 13 Pages
11 Problems
&
300 Minutes

A	Palinagram **Input:** Standard Input **Output:** Standard Output	

A *palindrome* is a word that is spelt the same backwards or forwards. Examples are "level" and "madam". An *anagram* is a word made from another word just by rearranging the characters, like "made" to "dame". For this problem *anagrams* and *palindromes* may not be valid English words. For example "daamm" is an *anagram* of "madam" and "amdma" is also palindrome although it is not an English word. "daamm" is also called a *palinagram* because it is the *anagram* of a *palindrome*. Any *palindrome* is also a *palinagram* but the vice versa is not always true. Some other examples *palinagrams* are "aabbcc", "aaaaa" etc. Given a string you will have to print the string that needs to be appended with it to make it a *palinagram*.

Input

The input file contains at most **6000** test cases. The description of each test case is given below.

Each case consists of a single string **S** of length **L** $(1 \leq L \leq 500)$. This string contains only lowercase English letters ('a' to 'z').

Input is terminated by a line containing a single hash ('#') character. This line need not be processed.

Output

For each line of input produce one line of output. The line contains the string that needs to be *appended* to the given string to make it a *palinagram (Can be an empty string as well)*. If there are more than one solutions output the shortest one, if there is still a tie output the lexicographically smallest one.

Sample Input	Output for Sample Input
ddc aaab #	 a

B Meeting Point of Circles

Input: Standard Input
Output: Standard Output

There are **N** circles. Each circle can move **1** unit distance per unit time (here distance is Euclidean distance). A circle moved **d** unit distance means its center moved a path (not necessarily in straight line) of **d** unit distance. When the center moves its body also moves with it (by body we mean circumference of course!). The circles can overlap each other. Our target is to provide all the circles enough time so that they can move themselves to such positions that all of their circumferences go through a common point. Find out the minimum time you need to provide them.

For example, suppose there are two circles one at co-ordinate **(0, 0)** and another at co-ordinate **(10, 0)**. Both of them have radius **1**. After **4** unit time first circle can move to **(4, 0)** and the second one to **(6, 0)**. Both of them go through the point **(5, 0)**.

Input

First line of the test file contains a positive integer **T** ($T \le 30$), number of test cases. Hence follow **T** test cases, each starting with a positive integer **N** ($1 < N \le 100$). Next **N** lines describe the circles of this test case. Each of these **N** lines contains three numbers **x y r**. **(x, y)** is the center and **r** is the radius of the circle. You may assume that **x** and **y** are integers having absolute value not greater than **1000**. **r** may be floating point number having at most **6** digits after its decimal point and it will not be greater than **1000**. No two circles of the input will be equal.

Output

For each test case, output the minimum time. Errors up to $5*10^{-4}$ will be ignored. It is safe for you to output at least four digits after the decimal point.

Sample Input	Output for Sample Input
2 2 0 0 1 10 0 1 2 1 0 10 -1 0 10	4.0000 0.0000

C	Dynamic accessible Pairs	
	Input: Standard Input **Output:** Standard Output	

C Dynamic accessible Pairs

Input: Standard Input
Output: Standard Output

Initially, there is an empty tree. You add **n** nodes to the tree, one by one.

after each node is added, print the number of accessible node pairs.

Two different nodes i and j are accessible if and only if **dist(i, j) ≤ r(i) + r(j)**, where **dist(i, j)** is the length of unique path from **i** and **j**.

Note that a node and itself is **NOT** an accessible node pair.

Nodes are numbered **1**, **2**, **3**, ... in the same order as they are added.

Input

The first line contains **n (2 ≤ n ≤ 100000)**, the number of total nodes.

There are **n** lines followed. The **i**-th line contains three integer **a(i)**, **c(i)**, **r(i)**, that means node **i** is connected with node **f(i) = a(i) XOR (last_ans mod 10^9)**, edge weight is **c(i)**, range value is **r(i) (1 ≤ r(i) ≤ 10^9)**.

Note that node **1** is not connected with any node, so we define **a(1)=c(1)=0**. For other nodes (i.e. **i ≥ 2**), **1 ≤ f(i)<i**, **1 ≤ c(i) ≤ 10000**, **0 ≤ a(i) ≤ 2*10^9**. For each test case, **last_ans** is initially **0**.

The input terminates with **n=0**, which should not be processed. There will be at most **5** test cases. In at most two cases, **n>10000**.

Output

The output for each test case contains **n+1** lines. The first line contains the case number, the **(i+1)**-th line is the number of accessible pairs after node **i** is added. Print a blank line after each test case (including the last one).

Sample Input	Output for Sample Input
5 0 0 6 1 2 4 0 9 4 0 5 5 0 2 4 5 0 0 6 1 2 4 0 9 4 0 5 5 0 2 4 0	Case 1: 0 1 2 4 7 Case 2: 0 1 2 4 7

D	# Gauss Reborn **Input:** Standard Input **Output:** Standard Output	

Do you know the childhood story of Carl Friedrich Gauss? He was of course the most talented kid in the class. Math teacher could hardly finish giving mathematics problem; Gauss was ready with his answer. One day being annoyed, the teacher asked Gauss to add numbers from **1** to **100**. The teacher was thinking that now he will be relieved from Gauss, cause adding **100** numbers is not an easy task. But guess what! Gauss was ready with his answer again. Anyway, I would not describe how he added all those numbers in such a short time but want to share a news. I think Gauss has reborn!

In my class there is a kid who is extremely talented. Like Gauss I hardly finish giving him problem. In **5** years age he can sum fractions! Can you imagine? You remember how to add fractions right? If you have two fractions say $\frac{a}{b}$ and $\frac{c}{d}$ then their sum is $\frac{ad+bc}{bd}$. Sometimes we divide numerator and denominator with a number to bring it to reduced form. Tomorrow I am going to give him a problem which is supposed to take a long time to solve. The problem is:

Given two numbers n and k. For $1 \le a, c, d, k \le n$ and $b = k$, how many tuples (a, b, c, d) are there such that $\frac{a}{b}$ and $\frac{c}{d}$ sum up to $\frac{ad+bc}{bd}$ and this sum is in reduced form. A fraction is in reduced form if the numerator and denominator have no common divisor other than **1**. Well, I don't want to take the risk, I will ask for sum of all **a**, sum of all **b**, sum of all **c** and sum of all **d** also. I hope he wont disturb me anymore within this week. But in case he turns in earlier I want you to help me to verify his answer.

Input

First line of the test file contains a positive integer **T (T ≤100)**. Hence follows **T** cases, each case starts with two positive integers **n** and **m**. **(n ≤ 100000 and m ≤ 100)**. Hence follows **m** space separated positive integers in the following line. These are the values of **k (k ≤ n)**.

Output

For each case, output the case number, followed by m lines. Each line contains five numbers the number of tuples, sum of **a**, sum of **b**, sum of **c** and sum of **d**. Since the answers may be big print them modulo **71211919**. If you are wondering why the mod is such peculiar, then I must say it is numerical form of Gauss.

Sample Input	Output for Sample Input
2 2 2 1 2 3 3 1 2 3	Case 1: 6 9 6 8 8 2 2 4 3 2 Case 2: 21 42 21 39 39 10 20 20 18 18 10 15 30 20 14

Explanation:
For n = 2 and b = k = 2:

a	c	d	$\frac{ad+bc}{bd}$	Valid
1	1	1	$\frac{3}{2}$	✓
1	1	2	$\frac{4}{4}$	✗
1	2	1	$\frac{5}{2}$	✓
1	2	2	$\frac{6}{4}$	✗
2	1	1	$\frac{4}{2}$	✗
2	1	2	$\frac{6}{4}$	✗
2	2	1	$\frac{6}{2}$	✗
2	2	2	$\frac{8}{4}$	✗

So 2 valid tupples. Sum of a = 2, Sum of b = 4, Sum of c = 3, Sum of d = 2.

E	Game of Throne Season 2 **Input:** Standard Input **Output:** Standard Output

Game of Throne Season 2

Input: Standard Input
Output: Standard Output

Once again, people of Gigaland are unhappy because of political unrest between Broken Arrow and Shadow Coder. This time Broken Arrow wants to be in one step ahead of Shadow Coder in terms of war strategy. He wants to send his army to the battlefield from Megaland as soon as possible.

For strategic reason, Broken Arrow wants to send at least **K** unit of army to the battlefield. The country of Gigaland consist of **N** cities (including Megaland and the battlefield) numbered from **1** to **N** and **M** unidirectional road. If Broken Arrow wants to send an army troop from city $\mathbf{U_i}$ to $\mathbf{V_i}$ using road **i**, the army troop will need $\mathbf{D_i}$ days to reach city $\mathbf{V_i}$ starting from $\mathbf{U_i}$ and at most $\mathbf{W_i}$ armies can start their journey on each day from city $\mathbf{U_i}$ to go through the road **i**.

As a great programmer of Gigaland, it is your duty to help Broken Arrow to find the minimum number of days required to send at least **K** armies from Megaland (city **1**) to the battlefield (city **N**).

Input

Input starts with an integer, **T** $(\mathbf{T \le 15})$ denoting the number of test cases. Each test case starts with three integers, $\mathbf{N\ (2 \le N \le 50)}$, $\mathbf{M\ (1 \le M \le N*N)}$ and $\mathbf{K\ (1 \le K \le 100000)}$. Each of the next **M** lines contain four integers $\mathbf{U_i}$, $\mathbf{V_i}$, $\mathbf{D_i}$ and $\mathbf{W_i}$ $\mathbf{(1 \le U_i, V_i \le N,\ U_i \ne V_i,\ 1 \le D_i, W_i \le 100000)}$, description of road **i**. There will always be a path from Megaland to the battlefield and also there can be multiple paths.

Output

For each test case, print the test case number, starting from **1**, and the minimum possible days required to send at least **K** armies from city **1** to city **N**. If the answer is more than **100** then print **-1**.

Sample Input

```
2
3 3 4
1 2 1 5
2 3 2 6
1 3 5 10
3 3 9
1 2 1 5
2 3 2 6
1 3 5 10
```

Output for Sample Input

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

F	Gift Dilemma **Input:** Standard Input **Output:** Standard Output	

Gift Dilemma

Input: Standard Input
Output: Standard Output

You all know Dr. Sheldon Cooper. Everything Sheldon gets, he must give back the same. He doesn't want to be in someone's debt.

Another friend of Sheldon, Penny has given him a very precious gift for his birthday. What penny didn't know, now Sheldon is going over his head to find out some gifts that will exactly match the price of Penny's gift. So my first task was to find out the price of the gift. With my super powerful sixth sense I found out that the price was **P.**

Now comes the hardest part. Sheldon needs to buy a gift which will cost exactly **P**. He went to the nearby shop for this. It is a crazy shop. Because it sells only three types of things: Rubik's cubes, mouth organs and chocolates. The prices of them are **A**, **B**, **C** accordingly. Now Sheldon faces a dilemma. What to buy and how many to buy? Because it turns out he can buy things costing **P** in various ways. For example if, **A** = 202, **B** = 203, **C** = 200 and **P** = 606, then are two different way to buy gifts costing exactly **P**. The first way is buying no Rubik's cubes, two mouth organs and one chocolate. The second way is buying three Rubik's cubes, no mouth organs and no chocolates.

Now Sheldon is asking for your help (because I am retired from contests). Given **A**, **B**, **C** and **P**, you need to find out in how many ways he can buy gifts costing exactly **P**.

Input

The first line will contain **T** **(T ≤ 100)**, the number of test cases. Next **T** lines each will have four integers, **A**, **B**, **C** and **P** **(0 < A, B, C, P ≤ 100000000)** and **C / GCD (A, B, C) ≥ 200**, *where* **GCD (A, B, C)** means the greatest common divisor of **A**, **B** and **C**.

Output

For each case print "`Case C: W`" (without the quotes), where **C** is the case number and **W** is the number of ways Sheldon can buy gifts.

Sample Input	Output for Sample Input
`1` `202 203 200 606`	`Case 1: 2`

G Query for Divisor-free Numbers

Input: Standard Input
Output: Standard Output

You are given an array of **N** integers: A_1, A_2, …, A_N. You have to process **Q** queries on this array, where a query will be a pair of integers (**L, R**).

For each query, you have to find the count of *Divisor-free* numbers in the number sequence **S**, where **S** = A_L, A_{L+1}, …, A_R. A number A_i from the sequence **S** will be called *Divisor-free* if there is no A_j **(i != j)** in **S** such that A_j is a divisor A_i.

Input

The first line of the input contains an integer **T (T ≤ 5)** denoting the number of test cases. The first line of each test case contains two integers **N** and **Q** ($1 \le N, Q \le 10^5$). The following line contains **N** space separated integers A_1, A_2, …, A_N where $1 \le A_i \le 10^6$. In each of the next **Q** lines, there will be two integers **(L, R)** representing a query $(1 \le L \le R \le N)$.

Output

For each test case, print the case number in the format "`Case X:`" (here, **X** is the serial of the test case). Then print **Q** lines containing the answer for each query.

Sample Input	Output for Sample Input
2 10 5 4 6 2 7 5 11 14 21 13 2 2 6 4 8 2 8 3 7 4 9 5 3 4 6 8 1 5 1 5 2 3 3 3	Case 1: 4 3 4 4 4 Case 2: 1 2 1

H	Palindromic Sums **Input:** Standard Input **Output:** Standard Output	acm

There are 10^9 cards lying on a table, where the **i-th** card has the value $\mathbf{i}(\mathbf{1} \le \mathbf{i} \le \mathbf{10^9})$ written on it. Alice picked **N** cards from those and then Bob also picked **N** cards from the remaining cards. They noticed two interesting properties:

- None of the cards picked by Alice or Bob has any palindromic value written on it
- The sum of values between any one card of Alice and any one card of Bob is always a palindromic number.

Your job is to find one possible selection of cards for both Alice and Bob for **N=4400**. A number is called palindromic if it spells same both forward and backward.

Input

This problem doesn't have any input.

Output

The first line of output should contain **N** space separated integers denoting the cards picked by Alice. The second line of output should also contain **N** space separated integers denoting the cards picked by Bob. You can print any possible solution. The printed numbers must be distinct and have values between **1** and $\mathbf{10^9}$ (inclusive). And also they should satisfy the two properties mentioned above.

The sample output shows one possible output when **N=2**. You need to find a solution for **N=4400**.

Sample Input	Output for Sample Input
	27 128 94 104

I	Minimum Sum	

Minimum Sum

Input: Standard Input
Output: Standard Output

You are given **n** integers $\mathbf{a_1, a_2, ..., a_n}$ and you have to find the sum of **f(i,j)** for all pair of **i** and **j** such that $\mathbf{1 \le i \le j \le n}$.

$\mathbf{f(i,j) = |m-a_i| + |m-a_{i+1}| + .. + |m-a_j|}$ where **m** = minimum of $\mathbf{a_i, a_{i+1}, ..., a_j}$.

|x| = absolute value of **x**.

Input

First line contains $\mathbf{1 \le T \le 10}$ test cases. Each test case contains two lines. First line contains an integer $\mathbf{1 \le n \le 50000}$ and second line contain **n** space separated integers. Absolute value of those **n** integers will be smaller than or equals to **50000**.

Output

Output a single line containing the sum. Please see output format for more information.

Sample Input	Output for Sample Input
1 5 1 2 3 4 5	Case 1: 35

J	# The Largest Circle **Input:** Standard Input **Output:** Standard Output	

You are given the Cartesian coordinates of four corners of a room which has the shape of a parallelogram. You will have to find the area of the largest circle that fits inside it.

Input

The input file contains **20000** test cases. The description of each test case is given below.

The input for each test case is given in a single line. This line contains eight integers x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4. **($-1000 \le x_1, y_1, x_2, y_2, x_3, y_3, x_4, y_4 \le 1000$)** These four integers actually denote that the coordinates of the four corners of the room are **(x_1, y_1), (x_2, y_2), (x_3, y_3)** and **(x_4, y_4)** (In clockwise or counter clockwise order). The room always has the shape of a parallelogram.

Input is terminated by a case where all eight values are zero.

Output

For each line of input produce one line of output. This line contains the area of the largest circle that fits in the given room. The area should be printed in the form **(a/b)*pi** (representing $\left(\frac{a}{b}\right)*\pi$), where **a** and **b** are relative prime integers and **pi** is the ratio of circumference and diameter of a circle. If it is not possible to express the result in this format then output a **-1** instead.

Sample Input	Output for Sample Input
0 0 10 0 10 10 0 10 0 0 0 0 0 0 0 0	(25/1)*pi

K	# Kiano The Clause! **Input:** Standard Input **Output:** Standard Output	acm

So Kiano's child Riano likes her father's long beard. She started calling him Candy Clause because this tall and bearded person likes to play Candy Crush and looks like angry Santa Clause. One day, Kiano, bored with easy challenges (!) of Candy Crush started a new and simple game with Fallen. Fallen and Kiano start playing on an **N X 5** board, called the game board. Initially all the cells of the board contains **0**. Then they ask Riano, the judge of the game, to show them an interesting board. Riano sets an **M X 5** board that is divided into **M** rows and **5** columns called interesting board. Each of the cells of the interesting board contains either **0** or **1**.

Now in his only turn, Fallen can change any number of **0**s to **1**s on the game board. After Fallen's move, Kiano, in his only turn, can change any number of **1**s to **0**s on the game board. If Kiano can make any **M X 5** sub board exactly the same as the interesting board set by the honorable judge Riano, Kiano wins. Otherwise Fallen gets the point equal to number of cells he changed in his turn.

Fallen wants to win the game at any cost and he also wants to win as many points as possible. Print the maximum possible point Fallen can win. If Fallen loses the game, print **-1** instead.

Input

The first line of the input contains an integer **T** **($T \le 50$)** denoting the number of test cases. Each of the following **T** cases starts with a pair of positive integers **N** and **M**, where **N** is the number of rows of the game board and **M** is the number of rows of the interesting board set by Riano. **N** and **M** are separated by space. Following this line there will be exactly **M** lines, each having **5** digits, **0** or **1** describing the board set by Riano. Note that, **$1 \le M \le 15$, $1 \le N \le 50$, $M \le N$**.

Please note that for **90%** of the cases, **$M \le 5$**.

Output

For each input, print the output in the format, `Case X: Y` here **Y** is the maximum number of point Fallen will win. If no matter how Fallen plays, he loses always print **-1**.

Sample Input	Output for Sample Input
3 1 1 11111 2 2 11111 11111 3 3 10111 11001 11111	Case 1: 4 Case 2: 9 Case 3: 14