HOSTED BY

DEPARTMENT OF CSE, BUET

BUET Inter University Programming Contest, 2016

Problems: A – K (19 pages including Cover)

11-Nov-16

Problem A
The Hater of Powers

It is the year 4016 and BUET is organizing yet another inter university programming contest. Contestants from all around the galaxy are gathering to attend this grand galactic event. The final count of the contestants tallied to an impressive value of $\boldsymbol{10^{18}}$.

The coordinator of the contest, Sir Kaizer Von Abaddon, distributed each of the 10^{18} contestants with their own unique identification number (ID) ranging from 1 to 10^{18}. But something really bothered Sir Kaizer; he has always hated perfect powers, which are integers that can be expressed as a positive integer power of another number. If a number has greater exponent he will dislike it more and *he always considers the representation for which the value of the exponent is the greatest* (for example: 64 can be represented as 64^1, 8^2, 4^3 and 2^6, but he will only consider the last one since that has the greatest exponent value). Therefore he decided to be a bit unfair to those whose ID is a perfect power.

On the day of the contest once all the 10^{18} participants arrived at the campus, they were asked to form a single line to get their belongings checked. At first the contestants stood at some random order but Sir Kaizer soon asked everyone to exchange their positions till the entire line was sorted according to the value of their ID's exponent representation, that is if two IDs are $\boldsymbol{a^p}$ and $\boldsymbol{b^q}$, $\boldsymbol{a^p}$ would move somewhere front of $\boldsymbol{b^q}$ if $\boldsymbol{p}$ is smaller than $\boldsymbol{q}$. Furthermore only if $\boldsymbol{p}$ and $\boldsymbol{q}$ are equal, $\boldsymbol{a^p}$ would move to the front of $\boldsymbol{b^q}$**,** if $\boldsymbol{a}$ is smaller than $\boldsymbol{b}$. Even though the contestant had no idea why they had to do it, they did it till there were no more position exchanges possible. This way Sir Kaizer ensured that the ones with ID he hated were behind in the line and had to wait more before their turn came and thus they suffered more, maybe they might even withdraw from the contest from getting tired for waiting so long! No one did though, even though they were a bit irritated.

For example: the participant with ID 100000 now has earlier position in the line to the participant with ID 64, since 100000 can be represented as 10^5, while 64 is represented as 2^6 and the former has lower exponent value. The sequence of IDs of the contestants in the line would now look something like this:

2, 3, 5, 6 … 4, 9, 25 … 100000 … 64 … 576460752303423488, 1

Now Sir Kaizer wonders which contestant appears at some particular position within the line. He asks you some positions and you have to tell him the ID of the participants who appear in those positions.

Input

First line of the input is $\boldsymbol{T}$ **(1 <=** $\boldsymbol{T}$ **<= 5000)** which describes the number of positions questioned by Sir Kaizer, followed by $\boldsymbol{T}$ lines. Each line contains an integer $\boldsymbol{P}$ **(1 <=** $\boldsymbol{P}$ **<=** $\boldsymbol{10^{18}}$**)** denoting the position in that line. Remember that the total number of contestant is always 10^{18}.

Output

For each question print a line with the format: "**Case N: X**", where **N** is question number and **X** is the ID of the contestant appearing on the position asked on the **N**$^{\text{th}}$ question.

Sample Input	**Sample Output**
4 1 3 1000000000000000000 999999999999999999	Case 1: 2 Case 2: 5 Case 3: 1 Case 4: 576460752303423488

Problem B
War in the Wizarding World

There is a deadly war going on between two countries (Bitland and Byteland) in the wizarding world. They are fighting with each other in a large forest. There are some strategically important places in the forest known as Hubs. They have numbered all the Hubs with numbers 1 to **n**. There are roads in the forest where each road connects two Hubs. There is at most one road between any two Hubs. The army of Bitland wants to reach Hub **n** starting from Hub **1** by travelling the shortest possible distance.

Unfortunately, the spies of Byteland came to know their plan, but they were not sure about the details of any specific path. So they did not want to take any risk and placed mines on any road which is a part of any shortest path from Hub **1** to Hub **n**. Now, Bitland also have spies working in Byteland, thus they also came to know about this trap. So, they decided to follow the shortest path which does not contain any of those roads.

The chief of army of Byteland could assume that their trap could be exposed to their enemy. He does not know anything about the path Bitland is going to follow after placing mines. Now, the army of Byteland already deployed all the mines. So, they wanted to destroy roads (where no mines were placed) by bombing such that there would not be any path from Hub **1** to Hub **n**. Bombs are very costly, so they want to destroy roads by bombing in such a way that the summation of those destroyed roads' length is minimum.

Input

The first line of input contains a single integer, **T (T <= 50)**. Then **T** test cases follow. Each case starts with the number of Hubs, **n (2 <= n <= 300)** and total number of roads, **m (1 <= m <=** 10^4**)**. Then **m** lines follow, where each line contains three integers, **u, v (1 <= u, v <= n) and w (1 <= w <=** 10^4**)**, indicating a road between Hub **u** and **v** with length **w**.

Output

For each case, output "**Case x: y z**" in a separate line, where **x** is case number, **y** is the length of the shortest path the army of Bitland is going to follow and **z** is the summation of the length of the roads Byteland army chief wants to destroy. If there is no path from Hub **1** to Hub **n** before or after deploying the mines, print **-1** as both of the values of **y** and **z**.

Sample Input	Sample Output

2 3 3 1 2 3 2 3 2 1 3 3 2 1 1 2 4	Case 1: 5 2 Case 2: -1 -1

Problem C
The Wall

A long time ago in a galaxy far, far away, there lived two groups of people named **Stormtroopers** and **Rebels** who were always fighting with each other. They lived in a planet **Tatooine** which was a flat 2 dimensional land. There were **N Stormtroopers** and **M Rebels** in **Tatooine**. You are given **N** pair of integers denoting the coordinate of **Stormtroopers** and **M** pair of integers denoting the coordinate of **Rebels.** They don't like each other.

So they want to build a wall between them. You can assume that it is always possible to build a wall between them. A wall is a line in the two dimensional plane. So you can always find such a line which can divide the plane into two separate part such that all the **Stormtroopers** are in one side of the wall and all the **Rebels** are on the other side of the wall. You have to find the wall that maximizes the minimum distance from any **Stormtrooper** or **Rebel** to the wall.

Input

First line of the input gives the number of test cases, **T** to follow.

For each test case, first line gives one integer, **N**.

Next **N** lines has a pair of integers $\mathbf{X1_i}$, $\mathbf{Y1_i}$ each denoting the coordinate of the **Stormtroopers.**

Next line gives one integer, **M**.

Next **M** lines has a pair of integers $\mathbf{X2_i}$, $\mathbf{Y2_i}$ each denoting the coordinate of the **Rebels**.

<u>Limits:</u> 1 <= T <= 35; 1 <= N + M <= 200000; $\mathbf{-10^7 <= X1_i, Y1_i, X2_i, Y2_i <= 10^7}$

Output

Output the maximum of the minimum distances from any of the points to the dividing wall (Acceptable error is 1e-6). Print output in the given format.

Sample Input	Sample Output
2	Case 1: 0.5

1 0 1 1 0 2 2 0 0 1 0 2 0 1 1 1	Case 2: 0.5

Problem D
Perseus and the New Challenge

The life of Princess Andromeda, the daughter of King Cepheus and Queen Cassiopeia of Ados, is in danger as she is captive inside the cave of the most atrocious monster of all time, the Kraken. Killing the Kraken is beyond the reach of any human. But the Ephors of the kingdom have finally discovered a way to weaken the Kraken. In some parts of the cave of Kraken is hidden some special kind of diamonds. If someone can collect the diamonds from the cave, then the Kraken can be sent into deep slumber with some magic spells. The higher the number of diamonds, the stronger the spell would be.

But the task is perilous. Nobody in the kingdom is brave enough to go to the cave. Finally, Perseus, the famous Greek hero comes to the rescue. The Ephors introduce the rules of collecting the diamonds to Perseus.

1. The cave is a 2D grid with **n** rows and **m** columns **(1 <= n, m <= 500)**.
2. Each cell contains some non-negative integer **k** or **-1**. If the value is **-1**, then the cell is blocked and Perseus cannot go through. Or else Perseus can visit the cell and collect **k** diamonds from the cell. **($0 <= k <= 10^9$)**
3. Perseus can start at any cell in the left border of the grid and travel until he finally stops at a cell in the right border.
4. During this trip, Perseus can only go **up/down/right** and cannot visit a cell more than once.
5. Both in the left and right border, Perseus can go up and down to maximize his collection.

Special Case:

1. When Perseus is at the top cell of a column, Perseus can still go up by the help of a dragon, which demands him to pay $\mathbf{C_i}$ amount of diamond for the $\mathbf{i^{th}}$ column, and in return the dragon will

teleport Perseus to the bottom cell of the same column and vice versa (i.e. from bottom cell to the top cell) provided that the topmost and bottommost cells are not blocked . It should be mentioned here that Perseus should have at least $\mathbf{C_i}$ amount of diamonds with him to take help of the dragon of $\mathbf{i^{th}}$ column. **($1 <= C_i <= 10^7$)**.

Can you help Perseus to calculate the maximum number of diamonds he can collect?

Input

Input starts with an integer **T (T <= 150)**, denoting the number of test cases. Each case starts with a blank line. The next line contains two positive integers, **n** and **m**. Each of the next **n** lines contains **m** integers **k** that fill up the grid (If **k** is **-1**, then the cell is blocked). The next line contains **m** space separated integers $\mathbf{C_i}$ denoting the cost the dragon would be demanding to teleport him from the top to the bottom or from the bottom to the top of the $\mathbf{i^{th}}$ column. Each case is separated by a blank line.

Output

For each case, output the case number followed by the maximum number of diamond Perseus can collect. If it is impossible to move to the rightmost column as described, print **'Trapped'** instead of the number. Please refer to the sample input output for correct formatting.

Sample Input	Sample Output
4 4 4 -1 4 5 1 2 -1 2 4 3 3 -1 -1 4 2 1 2 50 11 17 50 4 4 -1 4 5 1 2 -1 2 4 3 3 -1 -1 4 2 1 2 50 13 17 50 4 4 -1 4 5 1 2 -1 2 4 3 3 -1 -1	Case 1: 17 Case 2: 15 Case 3: Trapped Case 4: 23

4 -1 1 2 50 7 17 50 4 4 -1 4 5 1 2 -1 2 4 3 3 -1 3 4 2 1 2 50 50 50 50	

Problem E
Oh Functions

Let's define functions f and g as:

$$f(x) = 2 * (f(x-1) + g(x-1) - 1) * (f(x-1) + g(x-1) - 3) + 12 * f(x-1) - 13$$

$$g(x) = (f(x-1) + g(x-1)) * (f(x-1) + g(x-1) - 4) + 12 * g(x-1) + 7$$

And $f(0) = 2$, $g(0) = 1$.

Here, x is non-negative integers. For giving a value of x, find the value of f(x) and g(x).

Input

First line of the input is **T (T <=50000)**, then **T** test cases follows. Each case has only one line containing a positive integer **x** $(1 <= x <= 10^{14})$.

Output

For each test case print a line in "**Case I: F G**" format where **I** is case number and **F** is **f(x)** and **G** is **g(x)**. As **F** and **G** can be very big number, output them with modulo **1,000,000,007**.

Sample Input	Sample Output
4 1 2 3 4	Case 1: 11 16 Case 2: 1367 820 Case 3: 9564839 4784068 Case 4: 261293629 130666607

Problem F
A Boring Game

Alice and Bob are playing a game with a tree with n nodes. Alice always plays first, and the two players move in alternating turns.

Before the game starts, they pick a vertex as the root of the tree. After that, the game begins. The game's rules are as follows:

- In a single move, a player can pick a vertex ***u*** which is NOT the root of the tree. (if someone picks the root, the whole tree will be gone and they will be very upset!)
- After choosing ***u***, the player will keep the sub-tree under u and will delete every other vertices.
- ***u*** will be the root of the new tree and other player will make his/her move.
- The game continues until someone is unable to make a move (when there is only one vertex left).
- The person who made the last move loses the game.

You can assume that each player plays optimally, meaning they will not make a move that causes them to lose the game if some better, winning move exists.

Now Alice wants to know how many vertices are there in the tree such that if they pick it as the initial root, she would always win the game. Can you help her to find it?

Input

The first line has an integer **T (1 <= T <= 10)** which describes the number of test scenarios. Then **T** tests follow. Each test has one integer at the first line **N (2 <= N <= 100000)**. The next **N-1** lines each contains two integers **X, Y (1 <= X, Y <= N; X != Y)**. This means there is an edge between vertex **X** and vertex **Y**. The tree is valid.

Output

For each test case, print the case number at the first line as "**Case t:**" where **t** is the test case number, followed by the number of such vertices. See the sample input/output for more details.

Sample Input	**Sample Output**
1 3 1 2 2 3	Case 1: 2

<u>Explanation:</u>

- Suppose 1 is the root of the tree. In that case, the Alice can pick the sub-tree rooted at vertex number 2 in her first move. Then the only choice for Bob is to pick vertex 3. After that, the game ends. As Bob made the last move, Alice wins the game.

- Suppose 3 is the root of the tree. Again Alice will win the game by picking the sub-tree rooted at vertex 2 in her first move.

- Suppose 2 is the root of the tree. Now Alice can pick either vertex 1 or Vertex 3. In either way, the game will end as soon as Alice makes her first move and she will lose the game.

So Alice can win the game if vertex 1 or vertex 3 is picked as the initial root.

Problem G
Mario and Princess Peach

The world of Mario can be imagined as 2D grid of row **N** and column **M** containing a total of **N*M** cell. Mario starts from top-leftmost cell and Princess Peach is located at bottom-rightmost cell.

In each step, Mario can jump *at most* **P** cell right or *at most* **P** cell down, where **P** is the power of the cell Mario is located. Obviously Mario must not go beyond the grid. Otherwise he will fall off and die. Each cell has **V** points located in it. When Mario steps on a cell he gets the points located in that cell (If **V** is negative |**V**| points is deducted). As you are a programmer, you want to find out what can be the maximum collected points in a given grid if player plays in an optimal way and rescues the Princess.

Input

The first line contains the number of test cases, **T**. Then **T** cases follow. Each of the test case contains **N, M** in the first line. Then **N*M** numbers follow showing the power of each cell. After that another **N*M** numbers follow showing the points of that cell.
Huge input file, avoid slower input methods.

<u>Limits</u>: T <= 20; 1 <= N, M <= 1000; 1 <= P <= 1000; |V| <= 100

Output

For each test case, print a line "**Case x: y**" where **x** is the case number and **y** is the maximum points Mario can get while rescuing Princess Peach.

Sample Input	Sample Output
2 3 3 3 3 2 2 3 2 1 1 2 3 -10 6 -1 2 -8 4 3 2 3 3 3 3 2 2 3 2 1 1 2 3 -5 7 -1 2 -3 2 4 1	Case 1: 12 Case 2: 11

Problem H
Different Standings

The 2016 Olympic Games is over, but in a planet far away from earth, there is a similar competition going on. Athletes from not only all countries in that planet but also from neighboring planets take part in that prestigious competition. As the number of participants is huge and a large number of them have same level of expertise, there are lots of ties for a place. The judges have also noticed that because of lot of ties, there are sometimes very few places in the standing compared to the number of participants.

They are interested to know how many different standings can be created with **n** athletes taking part in a competition when there will be at least **r** athletes tied in every place and at most **k** places can be present in the standings. A standing X is different from another standing Y iff one of the following conditions are satisfied:

1. There are different number of places in the standings X and Y.
2. There is equal number of places in both standings, but there exists at least one place such that the set of athletes in that place in standing X is different from the set of athletes in the same place in standing Y.

For example, say *n=3*, *r=1* and *k=2*. Let us denote the athletes with integers from [1,...,n]. As *k=2*, there cannot be any 3rd place in any standing. Now, one of the 3 athletes may claim the 1st place (in 3 ways); or any two of them may tie for the 1st place (in 3 ways); (In both of these scenarios, the remaining 1 or 2 athletes have no other option than to be 2nd) or all of them can tie in the 1st place (1 way). The following standings can be formed: *({1, 2, 3}), ({1}, {2, 3}), ({2}, {1, 3}), ({3}, {1, 2}), ({2, 3}, {1}), ({1, 3}, {2}), ({1, 2}, {3})*. Therefore, the total number of standings will be *7*.

Input

The first line of input contains a single integer, **T ($T<=10^5$)**. Then **T** lines follow. Each of these **T** lines will contain *3* integers: **n ($2 <= n <= 10^4$)**, **r (1 <= r <= 20)** and **k (1 <= k <= 100)**.

Output

For each of the cases, print output "**Case x: y**" in a separate line, where **x** is case number and **y** is the number of different possible standings. As this number can be quite large, output **y modulo 1,000,000,007**.

Sample Input	Sample Output
3 3 1 2 3 2 2 2 3 1	Case 1: 7 Case 2: 1 Case 3: 0

Problem I
Gold Mines

The city of gridland is a grid with n rows and m columns. The city can be divided into n*m cells. A cell in is denoted by a coordinate (x, y) where x is the row number and y is the column number.

Two cells are adjacent if they share a side or a corner. A cell can have at most 8 adjacent cells. A path between two cells is defined by a sequence of adjacent nodes between the first cell and the last cell.

The scientists recently discovered gold in some of the cells. They marked the positions of all the cells which contain gold. Here is a sample map:

			G							G	G	G	G	G
		G	G	G					G		G			
		G	G	G				G			G			
						G	G	G				G	G	G
						G	G	G						
	G					G	G	G						
G														
G														
G							G	G						
G														
G														
G	G													
G	G													
G	G	G				G			G	G				
G	G	G	G	G	G	G	G	G	G	G	G			

In the figure above, all the cells marked “G” has gold in them. Let’s call the “gold-cells”.

Two gold-cells are part of the same goldmine if:

- There is a path between the cells
- All the cells which belong to the path contain gold.

In the figure above, there are 4 gold mines. Size of a gold mine is the number of gold cells in the mine. In the figure above, the top-left gold mine has size 7.

The government decided to select exactly two gold mines and connect them by a tunnel. A tunnel is simply a path between two cells. But to qualify as a tunnel, one end of the path must be adjacent to the first gold-mine, another end of the path must be adjacent to the second gold-mine. Also The path must not go through any gold-cells. (Note that a tunnel can have length 1)
The cost of building a tunnel = ***length of the tunnel * size of the first gold mine * size of the second gold mine***.

Size of a gold mine is the number of gold cells in the mine.

Your task is to minimize the cost of building the tunnel.

<table>
<tr><td></td><td></td><td></td><td>G</td><td></td><td></td><td></td><td></td><td></td><td></td><td>G</td><td>G</td><td>G</td><td>G</td><td>G</td></tr>
<tr><td></td><td></td><td>G</td><td>G</td><td>G</td><td></td><td></td><td></td><td></td><td>G</td><td></td><td>G</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td>G</td><td>G</td><td>G</td><td></td><td></td><td></td><td>G</td><td></td><td></td><td>G</td><td></td><td></td><td></td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td>||</td><td>G</td><td>G</td><td>G</td><td></td><td></td><td></td><td>G</td><td>G</td><td>G</td></tr>
<tr><td></td><td></td><td></td><td></td><td></td><td>||</td><td>G</td><td>G</td><td>G</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td></td><td>G</td><td></td><td></td><td></td><td>||</td><td>G</td><td>G</td><td>G</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>G</td><td></td><td></td><td></td><td></td><td></td><td>\\</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>G</td><td></td><td></td><td></td><td></td><td></td><td>||</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>G</td><td></td><td></td><td></td><td></td><td></td><td></td><td>G</td><td>G</td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>G</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>G</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>G</td><td>G</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>G</td><td>G</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>G</td><td>G</td><td>G</td><td></td><td></td><td></td><td>G</td><td></td><td></td><td>G</td><td>G</td><td></td><td></td><td></td><td></td></tr>
<tr><td>G</td><td>G</td><td>G</td><td>G</td><td>G</td><td>G</td><td>G</td><td>G</td><td>G</td><td>G</td><td>G</td><td>G</td><td></td><td></td><td></td></tr>
</table>

The second figure shows the optimal way to build tunnel for this map. The cost is 5 *7 * 2 = 70.

Given the configuration of the grid, print the minimum cost.

Input

The first line contains **T (1 <= T <= 120)** which describes the number of test cases. First line of each test cases contains two integers, **n** and **m (1 <= n, m <= 40)**. Each of the next **n** lines contains **m** characters denoting the grid.

Output

For each case, print the case number and the minimum cost. Print "**-1**" without the quotation marks, if there is **less than two** gold mines in the grid.

Sample Input

```
2
12 12
......GGGGG.
......G.....
G.....G.....
GG....G.....
GG....G.....
G...G.GG....
.G..GGGG....
....GGGG....
.....G......
.......G....
............
............
5 5
GGGGG
G...G
G.G.G
G...G
GGGGG
```

Sample Output

```
Case 1: 21
Case 2: 16
```

Problem J
Final Days of Goldfinger

We see agent Bond once again being called in time of a great need. The vile criminal Auric Goldfinger has stolen a metric tons of gold off the U.S military gold deposit right under their nose. Agent Bond has been tracking Goldfinger for a long time after the foul villain has made his mark by establishing the world's largest underground gold smuggling network, but this time Goldfinger has left enough clues to be tracked to his hideout. There is no saving for Goldfinger now! Once he finally faces agent Bond no matter how many bodyguard Goldfinger has hired and how many golden fortress he has made, nothing can stop from bringing him to justice.

Goldfinger has made his hideout in a large abandoned oil processing factory. The entire factory is composed of several structures, towers and such, lined in a straight line from left to right. Agent Bond doesn't really know which structure he will find Goldfinger in, but he is sure that agent Bond will always move from one structure's ceiling to the ceiling of the structure immediately to its right. Agent Bond plans to be dropped on one of the structure by a helicopter, from which he will keep moving right by jumping from ceiling to ceiling until he finds Goldfinger.

The structures are serially numbered, the leftmost being 1, where structure $\boldsymbol{i}$ have a height $\boldsymbol{H_i}$. Agent Bond will **lose** some potential energy to climb the next structure $\boldsymbol{i+1}$ if it's height $\boldsymbol{H_{i+1}}$ is greater than $\boldsymbol{H_i}$, and he will **gain** some potential energy if the next structure has lower height instead. The amount of energy he gains or losses is equal to the absolute difference in $\boldsymbol{H_i}$ and $\boldsymbol{H_{i+1}}$. Since agent Bond *starts with zero* energy, his energy might go below zero, so what happens when this does occur you might ask? Well, that's what Goldfinger would want too because Bond will die. But clever agent Bond is well aware of it, so what he can do is before he lands on the factory he will drink some of his favorite wine to **gain** some starting potential energy, enough to carry with him on his endeavor. Agent Bond doesn't want to drink too much wine though, just enough to get the energy to get from his starting position to his target structure. Let's call this minimum needed energy the optimum energy.

Now agent Bond has collected the list of height of all the structures in the factory and in the order that they currently are and plans to do some dry runs. Some of the heights are also negative, but you don't need to think much about it though since agent Bond knows what he is doing. He has also hired you to write a program that will do some tasks on the data and answer some of his queries. What agent Bond asks you to do is described below:

Task Type 1:

i h Which means he asks you to change the height of the $\boldsymbol{i}^{th}$ building to $\boldsymbol{h}$.

Task Type 2:

l r Now this one is a bit complicated! You have to tell him the maximum among all the optimum energy he would need for every possible ***i, j*** such that ***l* <= *i* <= *j* <= *r***, and he starts from the structure ***i*** and would end up finding Goldfinger in structure ***j*** without dying.

The feasibility of this mission now lays on your shoulder, my friend. Do not let agent Bond down or else you will disappoint the entire world.

Input

First line of the input is ***T*** **(1 <= *T* <= 10)** which describes the number of test cases. For each test case, the first line contains two integers ***N*** and ***Q*** **($1 <= N, Q <= 10^5$)**, where ***N*** is the number of structures in the factory and ***Q*** is the number of tasks agent Bond asks you to do. The next line contains ***N*** integers, i^{th} of which describes the height H_i **($-10^9 <= H_i <= 10^9$)** of i^{th} structure.

Then follow ***Q*** more lines, where each line contains three integers each. If the first integer is ***1***, use the next two integers ***i*** **(1 <= *i* <= *N*)** and ***h*** **($-10^9 <= h <= 10^9$)** to carry out the *Task Type 1*. If the first integer is ***2*** instead, use the next two integers ***l*** and ***r*** **(1 <= *l* <= *r* <= *N*)** to answer *Task Type 2*.

Output

For each test case, print a line with the format: "**Case x:**" where **x** is test case number; then, for each of the *Task Type 2* asked in the x^{th} test case, print the answer in a new line.

Sample Input	Sample Output
1 7 4 -10 6 5 -4 -3 8 -9 2 1 7 1 6 5 2 1 7 2 2 6	Case 1: 18 16 9

Explanation:

In the sample data we only deal with one test case. Here there are three tasks of type 2 and one task of type 1.

For the first task "2 1 7", there are total of 21 possible scenarios where agent Bond will start at some structure and end at some other. For example one of them would be where agent Bond starts from the 2nd structure and ends up in the last (7^{th}) one, and for this journey he would need an optimum energy of 2. But we can see that for the journey that starts from 1^{st} structure and ends at the 6^{th} he would need an optimum energy of 18 to reach the target, and this is the maximum among all the possible scenarios and thus is the answer to the first task.

Problem K
Kinderbijslag

Kinderbijslag (children's allowance) is a social security payment which is distributed to the parents or guardians of children, teenagers and in some cases, young adults. This benefit is meant to allay the expenses of care for a child, such as clothing, sports activities and so forth.

In the EU, the country responsible for your social security, including family benefits (child benefits, child-raising allowances and so on), depends on your economic status and your place of residence - not your nationality.

National laws determine the conditions under which parents are paid family benefits. Usually, parents are entitled to benefits in a given EU country:

- if they work there
- if they receive a state pension under that country's social security scheme (for example old-age, invalidity or survivor's pension)
- or simply if they live there.

Beware that family benefits differ greatly within Europe.

Diaper costs have a huge impact on the cost of having a baby. To figure out the cost of diapers you also need to know how long your baby will be wearing diapers. Some researchers say that the average age for completion of “potty training” for a girl is about 34-36 months and for a boy it’s about 37.5-38 months.

Given amount of Kinderbijslag you will receive per month for 1 child , average per unit diaper cost and amount of diaper needed per month, you have to determine whether this Kinderbijslag is enough to cover the cost of diaper or not.

Input

The first line of the input contains an integer **T (1 <= T <= 600)** denoting the number of test cases. Each test case contains 3 integers **M, P, N (0 <= M <= 500; 10 <= P <= 100; 150 <= N <= 240)** which indicate amount of kinderbijslag in EURO (€), per unit diaper cost in EURO CENT (1€ = 100 cents) and amount of diaper needed per month.

Output

For each test case, print "**YES**" if Kinderbijslag is enough to allay the cost of diaper. Otherwise print "**NO**". See sample input/output for details.

Sample Input	Sample Output
2 65 23 160 25 23 160	Case 1: YES Case 2: NO