# 2013-2014 CT S01E10: 2013 ACM-ICPC Egyptian Collegiate Programming Contest (ECPC 2013)

## A. Rasheda And The Zeriba

time limit per test: 2 seconds
memory limit per test: 256 megabytes
**input: zeriba.in**
output: standard output

Yesterday a strong storm hit the city, destroying many of the animals' stables. Unfortunately **Bakkar's** little cute goat (**Rasheda**) was not lucky, and her zeriba (stable) was also destroyed by the storm.

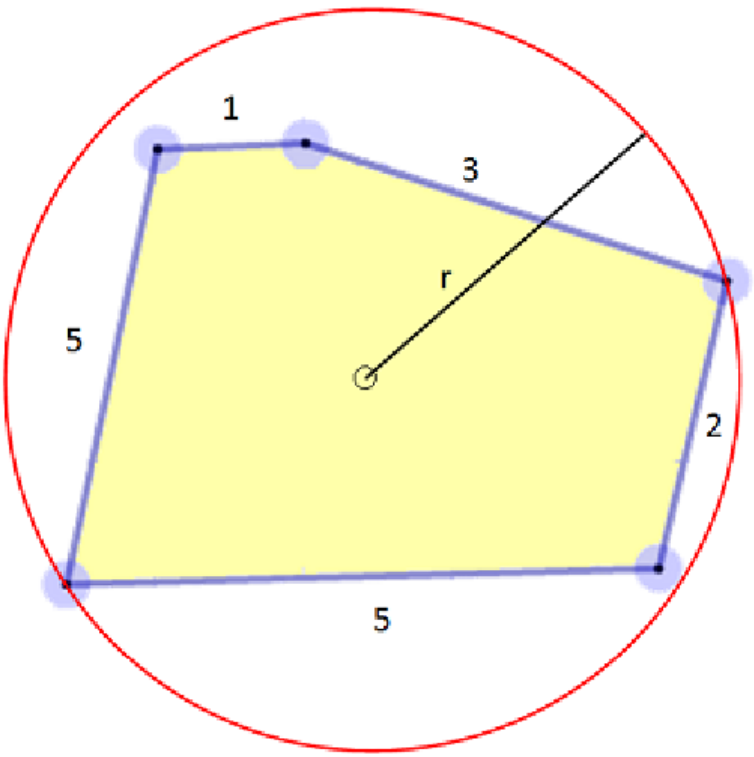
**Bakkar** is now searching for a better place for **Rasheda**, so he decided to buy a new piece of land to build **Rasheda**'s zeriba on it. The land owner only sells circular pieces of land. **Bakkar** has only a little amount of money to buy land and build the zeriba on it. Fortunately **Bakkar** managed to recover most of the wooden walls of the old zeriba, so he will use them to build the new one without buying new walls. To minimize the amount of money spent, **Bakkar** has to buy the smallest piece of land that can enclose the zeriba he is building.

To build the zeriba **Bakkar** uses the following strategy; using the ***N*** walls recovered **Bakkar** would construct a non-degenerate convex polygon to build the zeriba. For construction purposes the walls of the new zeriba must be in the same order as they were in the old one.

Now given the ***N*** lengths of the walls recovered, **Bakkar** will construct the zeriba so that it is a non-degenerate convex polygon with ***N*** sides, using the same order of the given walls. **Bakkar** then needs to know the radius of the smallest circle that can surround the entire polygon.

As you are a geometry lover, **Bakkar** called you to ask for you help. He will tell you the lengths of the walls then you would tell him the radius of the minimum circle that can enclose the zeriba if it is possible to construct a non-degenerate convex polygon using the sides. If it is not possible you would tell him that it's impossible.

P.S.: A convex polygon is a polygon that has no angles pointing inwards. More precisely, no internal angle can be more than $180^\circ$. A non-degenerate polygon is a polygon with a non-zero area.




The above figure illustrates a possible configuration of the first sample input in which there are 5 walls with lengths 5,5,2,3 and 1 respectively.

### Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer ***T***, the number of test cases ($1 \leq T \leq 100$). Followed by ***T*** test cases, each test case start with a single integer ***N***, the number of walls recovered ($3 \leq N \leq 1000$). Following this, there are ***N*** space-separated positive integers, representing the length of each wall in anti-clockwise order. The length of each wall will not be more than 1000.

### Output

For each test case print a single line containing "Case n:" (without the quotes) where n is the test case number (starting from 1) followed by a single space, then a single real number representing the radius of the smallest circle the zeriba can fit in. Each output must be rounded to the nearest 0.0001 and must be printed with exactly four digits after the decimal point. If you can't form a convex polygon from the given walls just print "can't

form a convex polygon" (without the quotes).

**Sample test(s)**

| input |
| --- |
| 2<br>5<br>5 5 2 3 1<br>3<br>5 1 1 |
| **output** |
| Case 1: 2.9039<br>Case 2: can't form a convex polygon |

# B. Egyptian Roads Construction

time limit per test: 15 seconds
memory limit per test: 256 megabytes
**input: road.in**
output: standard output

Welcome to Egypt!!! where you can wait for decades for the government to accomplish something useful!. The roads of the city are completely damaged and haven't been maintained for years. The people of the city have made several requests to pave and maintain the streets to the governor but there was no response. Lately the governor was changed and to satisfy the people of the city the governor promised to do his best to get the streets paving budget from the government. In order to request the budget an estimation for the cost must be conducted. This estimation can be done by a company but it will cost a lot of money, and moreover, the city will have to wait for the governor to request an approval from the government to get the company expenses. Taking this option will get the city into "the chicken or the egg" dilemma.

Being positive citizens; **Bakkar** & **Hassona** decided to conduct this estimation using voluntary efforts from the people of the city. They decided to make you in charge of the team responsible for the estimation as you are an expert in road construction techniques and have some background in graph theory.

The graph of the city can be described as an undirected graph consisting of ***M*** edges (representing the city's roads) and ***N*** nodes (representing the roads' intersections), where each edge connects exactly two nodes and every road has a pavement cost. To conduct the estimation; the government needs to know for every node: what is the cost of the mini-max path to every other node. A mini-max path between two nodes is a path that connects the two nodes and the maximum pavement cost on the edges of this path is minimum. The cost of a path is the maximum pavement cost among all its edges. The city roads guarantees that there is at least one path between any two nodes.

Since you are in charge of the team, you asked them to get you the pavement cost for all roads. And once you get this information you will write a program to calculate the required estimate.

To make your life easier, the governor agreed with the government that they will conduct the estimation if you provide them with a single cost for every node that represents the sum of the costs of the mini-max paths from that node to every other node. Formally, for every node ***i*** you have to provide the government with the sum of the costs of the mini-max paths between node ***i*** and every node ***j*** where $1 \leq j \leq N$ and $i \neq j$.

Your team has got the pavement costs for all the edges ready for you. It's now your job to provide the government with the values requested.

## Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer ***T***, the number of test cases ($1 \leq T \leq 100$). The first line of each test case will contain two integers ***N*** ($1 \leq N \leq 10^5$) and ***M*** (***N***-1 $\leq M \leq 10^5$), the number of nodes and edges respectively. The next ***M*** lines will each contain three integers ***x***, ***y*** and ***c*** representing an undirected road between node ***x*** and node ***y*** with pavement cost ***c*** ($1 \leq x,y \leq$ N) ($0 \leq c \leq 1000$). There will be no road connecting a node to itself and there will not be multiple roads between any two nodes.

## Output

For each test case print a single line containing "Case n:" (without the quotes) where n is the test case number (starting from 1). Then for each node ***i*** ($1 \leq i \leq N$) print on a single line the sum of the costs of the mini-max paths from node ***i*** to every other node in the city.

**Sample test(s)**

**input**

```
2
5 7
1 2 2
1 4 5
2 3 14
2 4 5
2 5 4
3 5 34
4 5 58
7 11
1 2 40
1 3 8
1 4 11
2 3 29
2 6 17
3 4 3
3 6 31
4 5 46
5 6 40
5 7 15
6 7 53
```

**output**

```
Case 1:
25
25
56
29
27
Case 2:
154
184
149
149
215
```

```
184
215
```

## Note

**Warning: large Input/Output data, be careful with certain languages.**

# C. Tomb Raiders

time limit per test: 15 seconds
memory limit per test: 256 megabytes
**input: treasures.in**
output: standard output

Egypt is full of hidden treasures (tombs, golden statues ...), and many thieves (called tomb raiders) are searching to steal them. To find the location of these treasures you need a good knowledge of the area. The police needed some honest young people to help them with guidance in exploring these areas.

**Mashrat** - the most dangerous thief in the city - wants to retire robbery and spend the rest of his life relaxing on a luxury island as he got old and tired from being wanted. He wants to make a big treasures robbery as his last mission that will secure money for his plans.

As we remember when the city needed to pave its roads; **Bakkar** & **Hassona** helped the governor to estimate the roads' pavement costs, leaving a very positive impression on the governor. Once the governor knew that the police needed some help from young people from the city, he didn't hesitate to introduce **Bakkar** & **Hassona** to the chief police officer. The chief police officer gave them a little introduction about the thieves they are chasing and the area they are planning to rob.

The area that **Mashrat** is planning to rob can be represented as a grid, with ***N*** rows and ***M*** columns. In the beginning, **Mashrat** starts at position ($p_i$, $p_j$) which means he starts at the cell in the $p_i$th row and $p_j$th column (all positions are 1-based).

**Mashrat** and his gang heard that the police is coming after ***S*** time units and they want to finish before they arrive. But thieves are lazy, they don't want to do any work unless it's too late. A thief will hire other thieves to search for him if there is more than 1 time unit left, and he will pay for them later a percentage of the treasures they bring back to him (don't worry, thieves never betray each other!). A thief at position $(i, j)$ with $t$ time units left ($t > 1$), will hire a thief for every grid cell $(x, y)$ that satisfies $|i - x| + |j - y| < t$, (i.e. Manhattan distance). Note that $(x, y)$ can be equal to $(i, j)$. Hiring thieves takes 1 time unit (1 time unit for all the thieves hired not for each), and original thief does nothing from now on, and waits for the hired thieves to return with the treasures in the end. There can be multiple thieves at the same cell working in parallel.

All the thieves including **Mashrat** follow the above behavior until they have 1 time unit left. At this time, they do not hire anymore, and quickly dig the ground looking for treasures. A thief standing on cell $(i, j)$ will dig up $G[i][j]$ worth of treasure (even if multiple thieves are on the same cell, each will get $G[i][j]$).

The Egyptian Police guided by **Bakkar** & **Hassona** arrived very fast and caught them before they left the site (**Mashrat** future plans are now all ruined up; they had to share the treasure on the spot, didn't they??). The police are suspecting that the thieves have hidden some of the treasures before they arrived and again asked for **Bakkar** & **Hassona**'s help. As computer geeks, **Bakkar** & **Hassona** are required to calculate the total value of stolen treasure by all thieves. Since the value can be very large, the police only want it modulo 1,000,000,007.

## Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer ***T***, the number of test cases ($1 \leq T \leq 100$). Followed by ***T*** test cases.

Each test case start with five integers separated by a single space ***N***, ***M***, ***S***, $p_i$, $p_j$, the number of rows, the number of columns, the time till the police arrives, and the position of **Mashrat**. ($1 \leq N,M,S \leq 100$, $1 \leq p_i \leq N$, $1 \leq p_j \leq M$).

***N*** lines follow, each with ***M*** non-negative integers, representing the values in ***G***. The values in each cell will not be more than 10,000.

## Output

For each test case print a single line containing "Case n:" (without the quotes) where n is the test case number (starting from 1) followed by a single space, then a single integer representing the total value of treasures stolen modulus 1,000,000,007.

**Sample test(s)**

| input |
| --- |
| 1<br>3 3 2 3 2<br>1 2 3<br>4 5 6<br>7 8 9 |
| **output** |
| Case 1: 29 |

## Note

Test Case Explanation:

At the beginning there is only one thief at cell (3,2) but since he still have 2 time units left, he hires 4 thieves at positions (3,1), (2,2), (3,2), (3,3) (where distance is < 2). He doesn't hire at position (4,2) because its outside the grid. Now there is only 1 time unit left so each thief digs up the treasures giving a total of G[3][1] + G[2][2]+ G[3][2] + G[3][3] = 7 + 5 + 8 + 9 = 29

# D. Bakkar And The Algorithm Quiz

time limit per test: 2 seconds
memory limit per test: 256 megabytes
**input: rooks.in**
output: standard output

Many years have passed … **Bakkar** is now a computer science student and its his third year at college. This year they study an Algorithm course where they enjoy learning new algorithmic and problem solving techniques. The course instructor loves to give them programming puzzles to help him discover talented and hardworking students.

Moreover the instructor is a chess maniac; he loves to make programming puzzles about chess. Today the instructor told the students that he is going to make a quiz with a bonus value of 10 points (which is very high by the way). The students were very excited and motivated to know the problem and of course to be able to solve it to get the bonus ;).

The problem description is as follows:

***Unlike the normal 8 by 8 chessboard, this game is played on a N rows by M columns chessboard. You are allowed to place some Rooks on the chessboard. A rook placed on the cell (i,j) will attack all the cells in the ith row and jth column.***

***You really hate some cells on this chessboard, and you want to super-attack these cells by rooks. For a cell to be super-attacked, both of it's row and column must be attacked by rooks. A rook super-attacks the cell directly beneath it.***

***The goal of the game is to use the minimum number of rooks to super-attack all of the cells you hate. Since you thought that task is too easy, you also decided to count the number of different ways to solve this game with the minimum number of rooks. Rooks are identical, and two ways are considered different if a position has a rook in one way but is empty in the other. Since this number can be very large, you only need to calculate it mod 1,000,000,007.***

Interesting problem isn't it?!! Well **Bakkar** is very excited and already has a solution in his mind for the problem.

Ok; the time is over now for the quiz and **Bakkar** is claiming that he has solved the problem successfully and you as a Teaching Assistant in the course musr prepare a program to validate the students' solutions. Let's validate **Bakkar**'s solution by running some test cases against his solution and recording it for evaluation by the validator.

## Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer **T**, the number of test cases ($1 \leq$ **T** $\leq 100$). Followed by **T** test cases, each test case starts with a line containing three integers separated by a single space **N**, **M** and **K** ($1 \leq$ **N**,**M** $\leq 10^9$, $1 \leq$ **K** $\leq 10^3$), representing the number of rows and columns in the chess board, and the number of cells you hate, respectively. The next **K** lines represent the cells you hate. Each line contains two integers separated by a single space **r** and **c** ($1 \leq$ **r** $\leq$ **N**, $1 \leq$ **c** $\leq$ **M**) which represent the represent the row and column of a cell you hate, respectively. All the cells will be distinct.

## Output

For each test case print a single line containing "Case n:" (without the quotes) where n is the test case number (starting from 1) followed 2 integers, space separated, representing the minimum number of rooks needed and the number of ways to place those rooks that **Bakkar**'s solution outputted.

**Sample test(s)**

| input |
| --- |
| 2<br>4 5 2<br>1 2<br>1 3<br>3 4 3<br>1 1<br>2 2<br>3 3 |
| **output** |
| Case 1: 2 7<br>Case 2: 3 6 |

# E. Ghanophobia

time limit per test: 2 seconds
memory limit per test: 256 megabytes
**input: ghanophobia.in**
output: standard output

It’s the first day of Eid Al Adha – Happy Eid :D – and all Egyptians are waiting for the fatal match that will decide if Egypt will qualify to the world cup 2014 or not. The qualification final step is made of 2 matches against Ghana. The first match will be played in Ghana and the second in Egypt. As we all know, the winner is decided by adding the results of the two matches, and in case of a tie, a team's goal at the homeland of its opponent is considered 2 goals. For example, if at the first match Ghana won with a result of 6:1, and in the second match Egypt won with a result of 5:0, then the total number of goals are considered equal (6:6). But as Egypt scores a goal in Ghana’s homeland the final result becomes 7:6 goals to Egypt and Egypt qualifies. Taking this fact in consideration, all Egyptians hope to make a positive score in Ghana to make their mission easier when playing the second match at Egypt.

And the match has begun and all Egyptians praying for their team. But the Egyptian team performance was disastrous; and the result was the most humiliating in the Egyptian’s team recent history. They were beaten with 1 goal to 6 for the Ghanaian team. The result and performance of the Egyptian team drove many Egyptian crazy. Many people were walking in the street hallucinating about the result of the second match that would make Egypt qualify to the world cup. Psychologists diagnosed these symptoms as **Ghanophobia**. To be brief; Ghanophobia symptoms can be summarized as the following: the patient starts asking whether a specific result for the second match would guarantee the qualification of the Egyptian team or not. For example a patient may ask 5:0 (5 goals for the Egyptian team to 0 for the Ghanaian team) and he should be answered with YES and for example if he said 1:0 he should be answered NO.

The second match is very close and still many Egyptians are hoping to make up the match. As many Egyptians **Bakkar** and his friends decided to support their team and go to the stadium to watch the match. But unfortunately the stadium will be full of people having Ghanophobia which will make it very hard to focus on supporting the Egyptian team while many people are hallucinating. **Bakkar** decided to make automatic answering machines that can answer people with Ghanophobia so they don’t disturb other supporters with asking. As a computer science student **Bakkar** will program these machines. He is going to write a program that given an expected result for the second match; it will answer with YES is this results would guarantee the qualification to the Egyptian team and NO if not. In case of tie print PENALTIES.

P.S.: Be careful while computing the results and don't say NO unless you are sure because if Egypt didn't qualify, Magdi AbdelGhany will nag us for 4 more years.

## Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer **T**, the number of test cases ($1 \leq \mathbf{T} \leq 10^4$). Followed by **T** test cases, each test case will consist of a string consisting of 2 number separated by a colon. The first number indicates the goals expected to be scored by the Egyptian team and the second number is the number of goals expected to be scored by the Ghanaian team. For example 6:1 means 6 goals scored by the Egyptian team to 1 goal scored by the Ghanaian team.

$0 \leq$ Scores $\leq 100$.

## Output

For each test case one line containing "Case n:" (without the quotes) where n is the test case number (starting from 1) followed by the answer for the given result.

**Sample test(s)**

| input |
| --- |
| 3<br>6:0<br>1:1<br>6:1 |
| **output** |
| Case 1: YES<br>Case 2: NO<br>Case 3: PENALTIES |

# F. Bakkar In The Army

time limit per test: 2 seconds
memory limit per test: 256 megabytes
**input: army.in**
output: standard output

**Bakkar** is now a senior college student studying computer science. And as many students; **Bakkar** fell in love with one of his finest colleagues **Maymona**. And as **Bakkar** has no brothers he is counting on getting an exemption from the military service after graduation. He got engaged to **Maymona** in their senior year counting on the exemption and a job he will get after graduation at the same place where he was interning last summer.

Well, man does not always get what he wants; the neither planned nor expected happened. **Bakkar**'s mother is pregnant and will give birth to **Hareedy** before **Bakkar** can get his exemption.

**Hareedy** is now born and unfortunately **Bakkar** will have to postpone his job and marriage plans for a year as he will serve as a military soldier for one year.

On the first 45 days, soldiers are trained in the military training center. They have to do a variety of exercises daily. One day **Bakkar** woke up late and didn't appear in the morning lineup at time. His commander is now angry and is going to punish him.

**Bakkar** is required to perform push-ups (the push-up position is called 6 esta'ed). His commander tells him to do them in reps (consecutive times) and then rest in between them. The commander wants him to follow a strict pattern. Given an upper limit, he will perform reps with increasing number of push-ups (1, 2, 3, ...) to warm up, until he reaches the upper limit. After that, he starts decreasing the number of push-ups per rep until he stops completely (..., 3, 2, 1). After resting, he will repeat the process again but with a higher upper limit. The upper limit starts with 1, and increases each time by a value of 1.

Here are the first 16 reps:

1

1 2 1

1 2 3 2 1

1 2 3 4 3 2 1 ....

The total number of push-ups he does is the sum of all the reps has has done so far. So for example, the total number of push-ups after completing 4 reps = 1+1+2+1 = 5, and after completing 7 reps = 1+1+2+1+1+2+3 = 11.

**Bakkar** now has to do at least ***N*** push-ups. This is very exhausting so he needs to know the minimum number of reps to complete using this pattern to reach his punishment reps.

## Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer ***T***, the number of test cases ($1 \leq T \leq$ 100,000). Followed by ***T*** test cases, each test case will be a single integer ***N***, the number of push-ups **Bakkar** wants to perform ($1 \leq N \leq 10^{18}$).

## Output

For each test case print a single line containing "Case n:" (without the quotes) where n is the test case number (starting from 1) followed by a single space, then a single integer representing the minimum number of reps needed as described above.

**Sample test(s)**

| input |
| --- |
| 5<br>6<br>9<br>11<br>21<br>35 |
| **output** |
| Case 1: 5<br>Case 2: 7<br>Case 3: 7<br>Case 4: 13<br>Case 5: 19 |

# G. Jenga In The Military Unit

time limit per test: 2 seconds
memory limit per test: 256 megabytes
**input: jenga.in**
output: standard output

After **Bakkar** finished the 45 days at the training center. He was assigned to the Air Defense Unit to complete his service period there. His day at the unit is very routine; he wakes up at 6 am to attend the morning line he starts exercising for about an hour. He has a daily guarding shift that lasts for 3 hours and varies from day to day in its timing. Then he starts to do some office work for the Major-General.

The day usually ends at 6 pm where he and the other soldiers rest for about 2 to 3 hours then they go to sleep. They usually take a shower, eat and then stay for a while talking or doing any boring stuff until they fall a sleep tired.

One day **Bakkar** and the other soldiers didn't have much to do on their day since most of the officers were not present at the unit. The day ended and they had some energy left. They were really bored and wanted to do something interesting so they decided to play some games but didn't know what to play. On his last vacation **Bakkar** saw an interesting game on Wikipedia called Jenga. He also played a couple of games with Hassona. He also had a picture of the game with him.

He started explaining the game to the other soldiers; Jenga is a two player game that starts with a tower of height ***N*** with 3 blocks on each level and the two players alternate turns. Consecutive levels have blocks perpendicular on each other. Each turn the player has to remove a block from a level that is fully covered from the top (i.e he can neither remove a block from the top most level nor from the level below it if the top most level is not yet full). Then he places the removed block in the top most level if it contains < 3 blocks, otherwise he creates a new level on the top of the tower using this block (making it perpendicular on the previous level as well). Newly added levels are treated like original levels, so players can take blocks from them while satisfying the mentioned restrictions. For the tower to maintain its balance every level must satisfy at least one of the following conditions:

1) The middle block exists in the level

2) The 2 side (outer) blocks exist in the level

Otherwise, the whole tower collapse. The players keep playing as long as there is a valid moves that won't make the tower collapse. The player who is not able to make a move loses the game.

The image above that **Bakkar** has, shows how the tower can look like in the middle of a game.

**Bakkar** and **Wahdan** will play against each other and they will use tower of height ***N***, **Bakkar** will play first and both play optimally. Can you find out who is going to win?

## Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer ***T***, the number of test cases ($1 \leq T \leq 10^4$). Followed by ***T*** test cases, each test case will be a single integer ***N***, the height of the tower ($2 \leq N \leq 10^9$).

## Output

For each test case print a single line containing "Case n:" (without the quotes) where n is the test case number (starting from 1) followed by a single space, then the name of the player who will win, either **Bakkar** or **Wahdan**.

**Sample test(s)**

**input**

```
3
2
12
100
```

**output**

```
Case 1: Bakkar
Case 2: Wahdan
Case 3: Bakkar
```

# H. The Job Interview

time limit per test: 15 seconds
memory limit per test: 256 megabytes
**input: dice.in**
output: standard output

**Bakkar** has just finished his military service. And as we know that he had a job offer from the same company he interned at when he was at college. But because of the delay caused by the military service the company wants to make sure that **Bakkar** is still a good candidate for the position. They know that **Bakkar** is very talented and a hard working person. That's why they didn't cancel the job offer and delayed his start date a year later. But they decided to conduct one final technical interview to ensure his readiness.

The interview had one problem solving question. The interviewer introduced himself and had a little conversation with **Bakkar** to update him with the company's state and to break any interview tension that **Bakkar** may have. Afterwards the interviewer introduced a problem to **Bakkar** on a whiteboard. The problem stated that a statistician was building a perfect random number generator machine using two 6-sided die (the plural of dice). He throws the 2 dice and the output is the sum of the top faces of each dice. The statistician uses any die as long as it has 6 distinct positive numbers on its faces.

The main problem lies that the statistician forgot the numbers on the faces of both die. But he can still know all the possible unique sums the machine can generate. **Bakkar** is now required to use this information to find the original values on the faces of both die. He will be given all the possible unique sums the machine can generate, and his task is to find any configuration for two die that will produce such sums if they are used by the machine.

**Bakkar** solved the problem and the interviewer is going to test it using some test cases he already generated. Do you think **Bakkar** will get the job and be able to marry **Maymona**?!! Let's see ;).

## Input

The program will be tested on one or more test cases. The first line of the input will be a single integer $T$, the number of test cases ($1 \leq T \leq 100$). Followed by $T$ test cases, each test case starts with a line containing a single integer $N$, the number of possible sums ($1 \leq N \leq 36$). The next line contains $N$ distinct positive integers, sorted in ascending order. Each integer $S_i$ represents one possible sum ($2 \leq S_i \leq 2000$).

## Output

For each test case print a line containing "Case n:" (without the quotes) where n is the test case number (starting from 1), followed by 2 lines. Each line contains the 6 values on the faces of one dice in any order. It is safe to assume that a solution always exists. If there are multiple solutions, print any.

**Sample test(s)**

| input |
| --- |
| 1<br>11<br>2 3 4 5 6 7 8 9 10 11 12 |
| **output** |
| Case 1:<br>1 2 3 4 5 6<br>1 2 3 4 5 6 |

# I. Bakkar In Zanzibar

time limit per test: 4 seconds
memory limit per test: 256 megabytes
**input: zanzibar.in**
output: standard output

Congratulations; **Bakkar** has just passed his interview and got the job. He is now going to call **Maymona**'s Father to arrange with him when they can make their wedding and finally get married. But wait a second, he just got an email from the company stating that he will have to travel to **Zanzibar** to complete a training on the project they are going to develop. Unfortunately **Bakkar** has to postpone his marriage for one more month. His trip will be next week and he is already packing up but this time he has already arranged everything with **Maymona**'s father and they agreed on the date of their wedding. By the way you are invited ;).

**Bakkar** has just arrived at the airport of **Zanzibar** and wanted to exchange some money for his transportation. He discovered a very strange thing about their money, they have banknotes only in the form of fractions of their currency **ZPound**. He asked **Koromba** (his trip guide) about that and he explained that all banknotes in **Zanzibar** are made of fractions of **ZPound** in the form ***a***/***b*** where $0 < a < b$ and $2 \leq b \leq 13$.

**Bakkar** arrived at the hotel to have some rest and to explore **Zanzibar** on the next day as it's a weekend in **Zanzibar**. **Bakkar** woke up and decided to go shopping to buy some gifts for **Maymona**.

While he was shopping he noticed that all prices are all less than 1 **ZPound**. He also noticed that some of the prices cannot be generated by the available banknotes. Later, he was informed that when the price cannot be generated by the banknotes, the convention is that the customer pays the amount of money that makes the minimum absolute difference to the original price (the buyer can actually pay more or less than the original price). Formally, if the price is ***X***/***Y*** **ZPound**, the customer will end up paying ***S*** **ZPound** such that |***S*** - ***X***/***Y***| is minimum. When the customer pays more than the original price, that is considered tips to the salesman. And when he pays less than the original price, that is considered a courtesy by the shop to the customer.

**Bakkar** got confused by that; so he decided to develop a mobile app to help him with this task. For a given price ***X***/***Y***, the mobile app should output the minimum absolute difference between what should be paid and the given price.

## Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer ***T***, the number of test cases ($1 \leq T \leq 10^5$). Followed by ***T*** test cases, each test case will be 2 integers ***X*** and ***Y*** representing an item that has a price of ***X***/***Y*** **ZPound**, where $X < Y$ and ($0 \leq X < Y \leq 10^5$).

## Output

For each test case print a single line containing "Case n:" (without the quotes) where n is the test case number (starting from 1) followed by a single space, then a fraction of the form ***a***/***b*** representing the minimum absolute difference between what should be paid and the given price where gcd(***a***,***b***) = 1.

The greatest common divisor "*gcd*" of two or more integers (at least one of which is not zero), is the largest positive integer that divides the numbers without a remainder. For example, the *gcd* of 8 and 12 is 4.

**Sample test(s)**

| input |
| --- |
| 3<br>13 17<br>33 47<br>11 21 |
| **output** |
| Case 1: 1/42840<br>Case 2: 1/109980<br>Case 3: 1/12012 |

# J. Anniversary Gift

time limit per test: 15 seconds
memory limit per test: 256 megabytes
**input: gift.in**
output: standard output

2 years have passed now since **Bakkar**'s graduation. He has completed his military service period and happily got married to **Maymona**. Next week is **Bakkar** & **Maymona**'s anniversary and fortunately **Bakkar** has a gift in mind for **Maymona**. He wants to buy her a new smartphone as hers was stolen on her way home in the metro.

**Bakkar** wants to make **Maymona** happy as much as he can. So a couple days ago he had a conversation with her about her preferable specs in a smartphone. He noticed that **Maymona** cares the most about the following four properties: height, width, thickness and weight.

**Bakkar** also has a very important fifth property which is the price of the smartphone. So **Bakkar** started to search the internet for the available smartphones. He was shocked by the variety of smartphones available (up to 10,000 phones). So **Bakkar** started filtering the phones list to get only the phones that satisfies the four properties that **Maymona** cares about while taking into consideration the price range. His goal now is to find how many smartphones match these criteria so he can take other hints from **Maymona** to further filter them. Given **Maymona**'s preference ranges for the 4 properties that she cares about, **Bakkar**'s price range that he can afford and a list of available phones each with its corresponding properties (height, width, thickness, weight and price), **Bakkar** as a Software Engineer will write a short program to filter the list of the smartphones to match **Maymona**'s and his preferences. The program will output the number of smartphones that matches their preferences.

## Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer ***T***, the number of test cases ($1 \leq T \leq 100$). Followed by ***T*** test cases. The first line of each testcase contains two integer ***N*** and ***Q***, the number of smartphones and the number of queries respectively($1 \leq N,Q \leq 10^4$).

***N*** lines follow, each line represents the properties of a single phone. Each one containing 5 floating point numbers ***p***, ***h***, ***w***, ***t***, ***we***, representing the price, height, width, thickness and weight. Each number will have at most 3 decimal places. ($0.000 \leq p, h, w, t, we \leq 5000.000$). For each test case, there is no more than 10 different values for each of the price, height, width, thickness and weight.

***Q*** lines follow, each line representing a query. Each one containing 10 floating point numbers ***p_low***, ***h_low***, ***w_low***, ***t_low***, ***we_low***, ***p_hi***, ***h_hi***, ***w_hi***, ***t_hi***, ***we_hi***, representing the lower and upper bounds of the properties. Each number will have at most 3 decimal places. ($0.000 \leq p\_low, h\_low, w\_low, t\_low, we\_low, p\_hi, h\_hi, w\_hi, t\_hi, we\_hi, \leq 5000.000$). $p\_low \leq p\_hi$; $h\_low \leq h\_hi$; $w\_low \leq w\_hi$; $t\_low \leq t\_hi$; $we\_low \leq we\_hi$.

## Output

For each test case print a line containing "Case n:" (without the quotes) where n is the test case number (starting from 1) followed by ***Q*** lines, each line contains a single integer, the number of smartphones that matches the query's preferences. A phone matches the query if and only if its value in every property lies between the lower and upper bounds of the query preference inclusive.

**Sample test(s)**

| input |
| --- |
| 1<br>10 3<br>5.0 6.0 6.0 5.0 4.0<br>1.0 9.0 3.0 8.0 0.0<br>1.0 7.0 7.0 5.0 5.0<br>4.0 0.0 1.0 9.0 8.0<br>6.0 0.0 5.0 6.0 0.0<br>6.0 7.0 6.0 5.0 4.0<br>1.0 4.0 5.0 6.0 6.0<br>5.0 7.0 8.0 9.0 8.0<br>1.0 3.0 3.0 7.0 9.0<br>1.0 2.0 8.0 4.0 8.0<br>0.0 1.0 2 3.0 4 5.0 6.0 7.0 8.0 9.0<br>4.0 0.0 1.0 5 0.0 6.0 7.0 6 9.0 8.0<br>3.0 2.0 5 7.0 8.0 5.0 5.0 6 8.0 9.0 |
| **output** |
| Case 1:<br>3<br>4<br>0 |

## Note

**Warning: large Input/Output data, be careful with certain languages.**

# K. Cubes Shuffling

time limit per test: 2 seconds
memory limit per test: 256 megabytes
**input: cubes.in**
output: standard output

It's Friday, and unfortunately the curfew starts at 7 pm today. **Bakkar** and **Maymona** are very bored as they won't be able to go out today. They decided to stay at home and try to do anything interesting to entertain themselves. They searched in their old toys boxes for puzzles or games they could play, interestingly they found some cubes that are numbered with random numbers. They started to shuffle the cubes boringly in a random way to see what numbers they can make.

**Maymona** then had an interesting idea, after shuffling the cubes in a row form, he started to sum up the sub-ranges of the cubes. for example if **Maymona** & **Bakkar** has 3 cubes numbered (1, 2, 3) they can then form 6 sub-ranges which are { [1], [2], [3], [1,2], [2,3], [1,2,3] }. Summing these sub-ranges { 1, 2, 3, 3, 5, 6 } we get a total sum of 20.

**Maymona** called this game **'Cubes Shuffling'** but **Bakkar** was not completely satisfied so he decided to make the game more interesting. He told **Maymona** they will take turns in shuffling the cubes and the first one who makes the shuffle with the minimum total sum of the sub-ranges would then be the winner.

**Maymona** & **Bakkar** started playing; and after many turns of randomly shuffling the cubes, Bakkar noticed that he can get the minimum shuffle in just one turn.

As you are a friend of **Maymona** you would like to help **Maymona** know the secret behind {Bakkar's} idea so you decided to write a program that given the cubes you would produce the arrangement that yields the minimum total sum of the sub-ranges. If multiple arrangements with the minimum total sum exists; you would produces the lexicographical smallest arrangement. An arrangement A is lexicographically smaller than an arrangement B with the same length if the first position they differ in is $i$ and $A_i < B_i$.

## Input

Your program will be tested on one or more test cases. The first line of the input will be a single integer ***T***, the number of test cases ($1 \leq T \leq 1000$). Followed by ***T*** test cases, each test case will consist of 2 lines. The first will contain a single integer ***N***, the number of cubes available in the toys boxes ($1 \leq N \leq 100$). The second line will contain ***N*** integers, representing the values written on the cubes ($1 \leq$ values $\leq 10^9$).

## Output

For each test case print a two lines. The first line contains "Case n:" (without the quotes) where n is the test case number (starting from 1). The second line will consist of the values in the minimum arrangement that can be produced, space separated.

**Sample test(s)**

| input |
| --- |
| 2<br>3<br>1 2 3<br>5<br>4 5 2 1 3 |
| **output** |
| Case 1:<br>2 1 3<br>Case 2:<br>4 2 1 3 5 |

# L. Mahdi And The Teddy Bear

time limit per test: 8 seconds
memory limit per test: 256 megabytes
**input: mahdi.in**
output: standard output

Time goes so quickly; remember **Bakkar**'s & **Maymona**'s anniversary two years ago when **Bakkar** bought a smartphone as a surprise gift for **Maymona**? Well **Maymona** had also a joyful surprise for **Bakkar**. **Maymona** was pregnant.

Months later **Maymona** gave birth to a beautiful baby and they decided to call him **Mahdi**. They were really so excited about their first child and decided to do their best to raise him up well. **Mahdi** is now about one year old and started to learn how to talk. As **Bakkar** & **Maymona** are geeks they had a very creative idea to teach **Mahdi** how to talk and evaluate his progress.

Simply they programmed a little teddy bear to speak so **Mahdi** can repeat after him. When **Mahdi** repeats what the teddy bear just said it recognises the word and analyse it. And as we all know its very natural that kids don't spell all letters at the beginning. So what **Bakkar** & **Maymona** thought to evaluate the correctness and progress of what **Mahdi** says is to check if the work **Mahdi** repeats after the teddy bear is a subsequence of the original word the teddy bear just said. To evaluate **Mahdi**'s progress the teddy bear defines a subsequence of a string **y** if it can be produced by deleting some of the characters in **y**. For example, "ace" is a subsequence of "adcbe" but not a subsequence of "bcae". If **Mahdi** spells a subsequence it is considered to be a positive indicator that **Mahdi** is progressing. To motivate **Mahdi** when he spells something correctly according to what we mentioned; if he spells a subsequence of the original word the teddy bear would respond with 'bravo' and start clapping, Otherwise he would repeat the word again several times until so **Mahdi** can take his time practicing.

The teddy bear is programmed in a very simple and creative way. He says random words based on an equation that **Bakkar** & **Maymona** invented. They specify a starting character to start the word, the length of the word, adder and multiplier. Simply the word is generated as follows:

$s_0$ = starting character

$s_i$ = char((pos($s_{i-1}$) * multiplier + i * adder) % 26)

Where:

pos(a) = 0, pos(b) = 1, ....pos(z) = 25

char(0)= a, char(1)= b,.....char(25)= z

Hmmmmm, **Bakkar** & **Maymona** are so busy; **Bakkar** at his work and **Maymona** taking care of the house and **Mahdi**. And as you are one of their best friends and still single and have some spare time you proposed to program the teddy bear for them. Besides you are the voice recognition geek among their friends ;).

For simplicity the words **Mahdi** repeats will be generated using the same equation the teddy bear uses.

## Input

The first line will be an integer T ($1 \leq T \leq 10$) the number of test cases. Each test case will start by a line containing 1 character and 3 integers (length, multiplier, adder) for the string ***S*** as described above ($1 \leq$ length(S) $\leq 10^7$). The next line will contain a single integer ***N*** ($1 \leq N \leq 10^4$) the number of strings **Mahdi** repeats. The next ***N*** lines each contain 1 character and 3 integers (length, multiplier, adder) for the string $L_i$ as described above. The starting character will be a lowercase character, ($1 \leq$ length($L_i$) $\leq 100$),

$1 \leq$ multiplier, adder $\leq 1000$.

## Output

For each test, For each test case print a single line containing "Case n:" (without the quotes) where n is the test case number (starting from 1) followed by N lines each containing either "BRAVO" (without quotes) if $L_i$ is a subsequence of S and **Mahdi** is making a progress or "REPEAT" (without quotes) otherwise.

**Sample test(s)**

| input |
| --- |
| 1<br>a 100 13 37<br>3<br>a 11 11 311<br>b 9 13 22<br>d 5 31 12 |

| output |
| --- |
| Case 1:<br>REPEAT<br>REPEAT<br>BRAVO |

## Note

for input d 5 31 12, the string will be

$s_0$ = d

$s_1$ = char((pos(d) * 31 + 1 * 12)%26) = char((3 * 31 + 1 * 12)%26) = char(105%26) = char(1) = b

$s_2$ = char((pos(b) * 31 + 2 * 12)%26) = char((1 * 31 + 2 * 12)%26) = char(55%26) = char(3) = d

$s_3$ = char((pos(d) * 31 + 3 * 12)%26) = char((3 * 31 + 3 * 12)%26) = char(129%26)= char(25) = z

$s_4$ = char((pos(z) * 31 + 4 * 12)%26) = char((25 * 31 + 4 * 12)%26) = char(823%26) = char(17) = r

so the string is "dbdzr"

**Warning: large Input/Output data, be careful with certain languages.**

---