IUBAT 2018 NCPC

IUBAT NATIONAL COLLEGIATE PROGRAMMING CONTEST 2018

Onsite contest: Aug 3-4, 2018

HOSTED BY

Department of Computer Science & Engineering
IUBAT-International University of Business Agriculture and Technology

IN ASSOCIATION WITH

ICT DIVISION
FUTURE IS HERE

DIGITAL BANGLADESH

Problem A

Input: Standard Input
Output: Standard Output

Alice and Bob

Have you heard about Bob and Alice? Game theory would have been incomplete without Bob and Alice. They can't spent a single day without challenging each other and playing different types of games.
Today, Alice and Bob started to play with an array consisting of **N** elements. The value of i^{th} element is denoted by A_i . Details about the game is as follows:

- Alice will randomly select an integer **X** and will ask Bob to remove those elements from the array where **($A_i \leq X$)**.
- It may divide the array into one or more segments. A segment is one or more consecutive non zero elements present in the array. The beauty of array will be the summation of beauty of all its segments.
- Beauty of a segment consisting of **k** elements = $H_{min} \times k$, here, H_i is the value of i^{th} element of that segment and $H_{min} = Min(H_1, H_2, \ldots, H_k)$.
- Bob needs to answer "**What will be the beauty of the array if he remove those elements from the array where $A_i \leq X$**".
- Effect of one query will have no longer effect on another.

For example:
Let, **N = 8** and $A_1, A_2, \ldots, A_n$ are 9, 10, 6, 8, 10, 5, 12 and 13 respectively.
If **X = 7**, then there will be three segments (9,10), (8,10), (12,13). So, total beauty of the histogram will be, $(9 \times 2) + (8 \times 2) + (12 \times 2) = 58$.
If **X = 8**, then there will be three segments (9,10) , (10) and` (12,13). So, total beauty of the histogram will be, $(9 \times 2) + (10 \times 1) + (12 \times 2) = 52$.

Input

Input starts with an integer **T ($1 \leq T \leq 20$)** denoting the number of test cases. First line of each test cases consists of two integers **N ($1 \leq N \leq 30000$)** and **Q ($1 \leq N \leq 30000$)** separated by a space. Next line contains **N** space separated integers $A_1, A_2, \ldots, A_n$ **($1 \leq A_i \leq 10^9$)** denoting the value of i^{th} element in the array. Each of the next **Q** lines contain an integer **X** where **($1 \leq X \leq 10^9$).**

Output

For each of the **Q** questions Bob needs to print the beauty of the array. Please see the sample output for more clarity. **N.B: Data Set is huge. Use faster I/O methods.**

Sample Input

```
1
8 6
9 10 6 8 10 5 12 13
9
8
7
4
5
10
```

Output for Sample Input

```
Case 1:
44
52
58
40
54
24
```

Problem B

Input: Standard Input
Output: Standard Output

Bipartite Permutation

Given a positive integer **N**, consider any permutation of all the numbers from **1** to **N**. It is required to create two partitions, $\mathbf{P_1}$ and $\mathbf{P_2}$, from these numbers such that $\mathbf{|sum(P_1) - sum(P_2)|}$ is minimum, where **sum(X)** denotes the summation of all the numbers in partition **X**. A partition is defined to be a non-empty subset of the permutation. In other words, find the minimum absolute difference between the summation of all the numbers in each partition. Note that you cannot leave out any number, every number from **1** to **N** must be part of exactly one partition.

Wait a moment! On second thoughts, this seems way easier than even the giveaway problem. To add a little bit more of spice, also find the lexicographically smallest partition $\mathbf{P_1}$ such that $\mathbf{|sum(P_1) - sum(P_2)|}$ is minimum. To make your life easier (or harder), you don't need to output the entire sequence, only the hash of the sequence as computed by the function below would suffice.

```
def sequence_hash(sequence, B, M):
    result = 0
    for number in sequence:
        result = (result * B + number) % M
    return result
```

Input

The first line contains an integer **T (1 ≤ T ≤ 1000)**, denoting the number of test cases. Each of the next subsequent **T** lines contain three positive integers, **N** ($\mathbf{2 \le N \le 10^9}$), **B** ($\mathbf{N < B \le 10^9}$), and **M** ($\mathbf{1 \le M \le 10^9}$).

Output

For each test case, first output the case number, followed by the minimum absolute difference and the hash of the lexicographically smallest partition $\mathbf{P_1}$.

Sample Input

```
12
2 10 1000000000
3 10 1000000000
4 10 1000000000
5 10 1000000000
6 10 1000000000
7 10 1000000000
8 10 1000000000
9 10 1000000000
1000 1000000000 1000000
1000000 1003001 998244353
123456789 987654321 666666667
444444444 666666666 888888888
```

Output for Sample Input

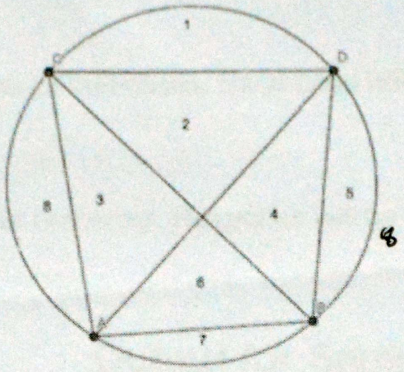
```
Case 1: 1 1
Case 2: 0 12
Case 3: 0 14
Case 4: 1 124
Case 5: 1 1234
Case 6: 0 1247
Case 7: 0 12348
Case 8: 1 123457
Case 9: 0 1000
Case 10: 0 553178755
Case 11: 1 214459309
Case 12: 0 557434257
```

Problem C

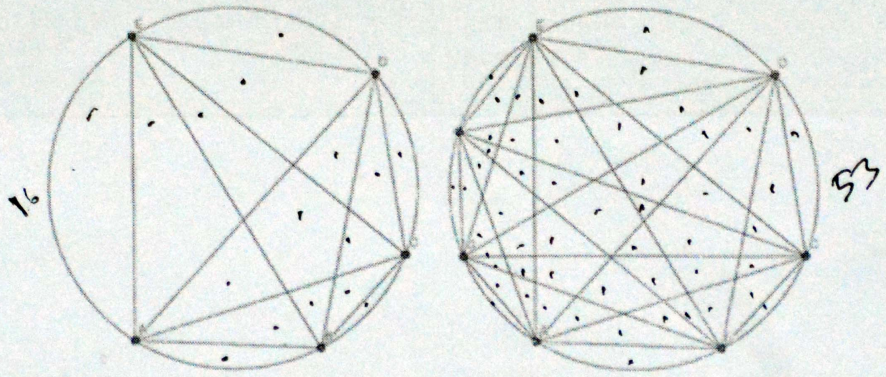
Input: Standard Input
Output: Standard Output

Circle Division

Given a circle with **N** points on its circumference, what is the maximum number of regions that we can get if we connect all the points with each other using a straight line? You can put the **N** points in any position of the circumference. For example, for **N = 4** we get maximum **8** regions:



Similarly for **N = 5** and **N = 7**, the divisions may look like below:



The above problem was set independently by me, but then **SM** pointed out that he has set the same problem years before. I was so sad, that I could not give you this wonderful problem. So, now I am trying to use the same problem in a different way.

The solution of the above problem uses Euler's famous formula. Euler's formula states that if a finite, connected, planar graph is drawn in the plane without any edge intersections, and **v** is the number of vertices, **e** is the number of edges and **f** is the number of faces (regions bounded by edges, including the outer, infinitely large region), then

$$v - e + f = 2$$

So using this formula and some combinatorics we can find that the number of vertices, edges in our circle division problem.

$$v = N + {}^{N}C_4$$

$$e = N + {}^{N}C_2 + 2({}^{N}C_4)$$

If we calculate **f** using the formula given above, we will find the number of regions. But it will have one extra region (the outside one). So using this method, the first few answers for different values of **N** is given below:

- N = 2, Answer = 2
- N = 3, Answer = 4
- N = 4, Answer = 8
- N = 5, Answer = 16
- N = 6, Answer = 31
- N = 7, Answer = 57
- N = 8, Answer = 99

So, you can see the pattern, that there are some values of **N**, for which answer is a power of **2**. And in other cases it's not. Now, given **N**, you have to figure out whether the answer would be a power of **2** or not.

Input

First line will contain **T** (**T < 10001**), number of test cases. Each case will contain an integer **N** ($1 < N \leq 10^4$).

Output

For each case, print case number and your response. Response will be "**Yes**" or "**No**" based on whether for the **N** (in input) the answer is a power of 2.

Sample Input	Output for Sample Input
3 4 5 7	Case 1: Yes Case 2: Yes Case 3: No

Problem D

Input: Standard Input
Output: Standard Output

Yet Another Closest Path Problem

We all know about closest path problem, where we have to find a path between two nodes in a graph such that the sum of the weights of its constituent edges is minimized. A path in a graph is a sequence of zero or more edges which connect a sequence of nodes(note that for this problem same node can appear multiple times in the path). For this problem, we don't want the closest path but we are actually interested to find the path with sum of the weights closest to a predefined value.

More specifically, you will be given an undirected weighted graph with **N** nodes numbered **1** to **N**. Then you will be given a start node **S**, an end node **F** and a value **C**. You have to find a path from **S** to **F** where the sum of the weights of the edges in the path is closest to **C**.

Input

First line of the input is **T (1 ≤ T ≤ 10)**, then **T** test cases follows. First line of a case will be two integers **N, M (1 ≤ N ≤ 400, 1 ≤ M ≤ 500)**. Next **M** lines each will have three integers **U**, **V** and **W (1 ≤ U, V ≤ N, 1 ≤ W ≤ 200)**. This will represent that in the graph there is an undirected edge from **U** to **V** with weight **W**. In the next line of the input will be one integer **Q** represents the number of query. Next **Q** lines each will have three integers, **S**, **F** and **C (1 ≤ S, F ≤ N, 1 ≤ C ≤ 1000)**. Sum of **Q** across all the test cases will be less than or equal to **500**.

Output

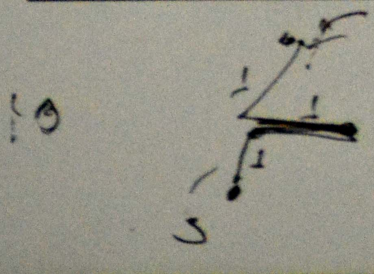
For each test case print a line in "Case t:" format where **t** is case number. Then in next **Q** lines for each query print the minimum difference between **C** and sum of the weights of the edges in any path of going **F** to **S** or **-1** if there is no path.

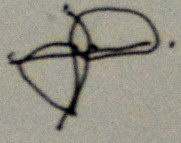
Sample Input

```
1
5 5
1 2 1
2 3 3
3 5 6
5 4 1
3 4 2
5
1 3 1
1 3 5
1 3 6
1 3 7
1 4 14
```

Output for Sample Input

```
Case 1:
3
1
0
1
0
```





Problem E

Input: Standard Input
Output: Standard Output

String is Easy

Everyone of us knows that string processing is very very easy in C++ using string container. We can easily figure it out how many times a string **P** contains another string **Q** as a substring in it. For example: if **P** = **aabaaa** and **Q** = **aa**, then **P** contains **Q** for exactly **3** times and this result can be calculated using C++ string container within few lines of code. That's why we were thinking of doing some string manipulation tasks. But many failed in this. Can you guys help us out from this situation?

You will be given a string **S** and another array of strings **A** containing **N** strings. Now, you will be given **M** queries containing **five** integers: L_S, R_S, L_A, R_A, **X**. Now you have to find how many strings between $A[L_A]$, $A[L_A+1]$, ... , $A[R_A-1]$, $A[R_A]$ (1-indexed) contains $S[L_S \dots R_S]$ (1-indexed) string exactly for **X** times.

Input

The first line of the input file contains an integer **T (1 ≤ T ≤ 3)** denoting the number of test cases. For each test case, the first line contains a string **S (1 ≤ |S| ≤ 100001)**. Second line of the test case contains the integer **N (1 ≤ N ≤ 100001)**. Each of the next **N** lines contains a string which are the elements of the array **A (1 ≤ $|A_i|$ ≤ 100001)**. Next line of the test case contains **M (1 ≤ M ≤ 100001)** denoting the number of queries. Each of the next **M** lines contains five integers L_S, R_S **(1 ≤ L_S ≤ R_S ≤ |S|)**, L_A, R_A **(1 ≤ L_A ≤ R_A ≤ N)**, **X (1 ≤ X ≤ 100001)** which together represents a single query. For each test cases, the summation of the length of all the strings will not exceed **510000**. Each string in the input file will contain only small letters.

Output

For every test case, the first line should contain the case number in the format: "Case X:" (without quotes) where **X** is the test case number. After that each of the next **M** lines should contains the answer of the queries.

Sample Input	Output for Sample Input
1 abaccdea 3 aba ea baa 3 1 1 1 3 2 3 3 2 3 2 2 3 1 3 1	Case 1: 2 1 2

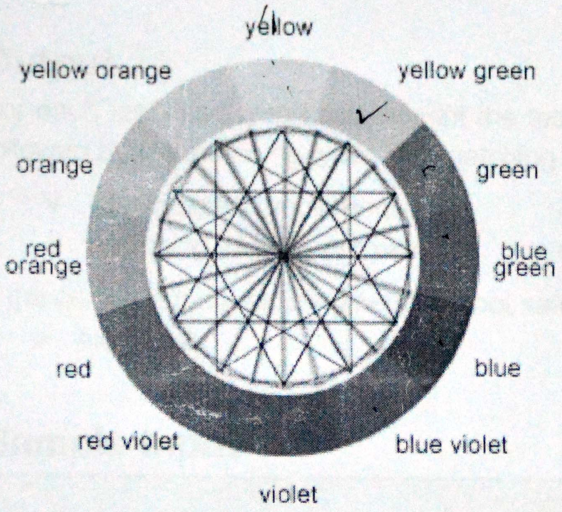
Problem F

Input: Standard Input
Output: Standard Output

Fashionable Nerd

Mr. Atel Ali is a very successful programmer. He was quite the nerd as a student. However, after a very recent job switch to the corporate world, he is finding himself out of place due to his lack of fashion sense. He just doesn't know how to match a clothing item with another in terms of color. He picks a light blue shirt but can't decide what color necktie to wear with the shirt.

Atel wants to pick the right combination of four items: 1) shirt, 2) pants, 3) shoes, 4) necktie. He can describe the color of each item in a combination of primary colors -- red, green and blue. There are five well known and rather easy to follow matching patterns. These patterns are based on the idea of arranging colors on a color wheel, like the one shown in the picture below.

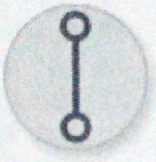


Given any color in RGB (red, green and blue) model, we can place it on the wheel depending on the hue angle. The hue angle, **H** can be computed as follows:

$$H = \begin{cases} \theta & \text{if } B \le G \\ 360 - \theta & \text{if } B > G \end{cases}$$

$$\theta = \cos^{-1}\left\{ \frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2+(R-B)(G-B)]^{1/2}} \right\}$$

The θ value (calculated in degrees) can then be evenly discretized to fall in any of the **12** sectors denoting **12** colors as shown on the color wheel. These **12** colors can be indexed from **0** to **11**. The index can be calculated as $\left\lfloor \frac{\theta}{30} \right\rfloor$.

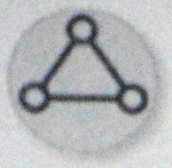


Complementary
The clothing items use only two different colors and the colors are from the opposite sides of the wheel. For example "yellow" shirt with "violet" pants, "violet" shoes and "violet" necktie.

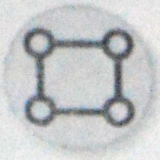
Monochromatic
The colors of all the clothing items are pretty much the same. For example, "green" shirt with "green" pants, "green" shoes and "green" necktie.

Analogous
It's a 3-color palette (only 3 different colors are used) where the colors picked are in a continuous sequence. For example, "yellow green" shirt with "green" pants, "yellow" shoes and "yellow" necktie.

4



Triadic
Another 3-color palette, but this time the colors need to be evenly spaced out in the color wheel. For example, "blue" pants, "red" shirt, "yellow" necktie and "red" shoes.



Tetradic
It's a 4-color palette where the colors are evenly spaced out in the color wheel. An example can be, "blue violet" shirt, "green" pants, "yellow orange" shoes and "red" necktie.

Note that, it does not matter which item uses which color.

Input

The first line of input contains the number of test cases, **T** (**T ≤ 1,000**). Each test case is written in exactly 4 lines; each line describes the color of a piece of clothing. Colors are expressed as RGB (red, green, blue) values. Note that, **0 ≤ R, G, B ≤ 255** and there is no such case where **R=G=B**.

Output

For each test case, you need to print the test case number (as shown in the output for sample input section), followed by the matching type. The matching type can be any of the following:

- Matching - Complementary
- Matching - Analogous
- Matching - Tetradic
- Matching - Monochromatic
- Matching - Triadic

If the colors of the 4 clothing items do not satisfy any of the matching patterns, the output should be:

- No match

Sample Input

```
4
183 116 51
183 171 51
51 183 89
51 133 183
183 145 51
183 155 51
183 159 51
189 161 87
189 96 87
189 117 87
189 87 115
189 87 181
132 125 58
132 119 58
132 113 58
58 80 132
```

Output for Sample Input

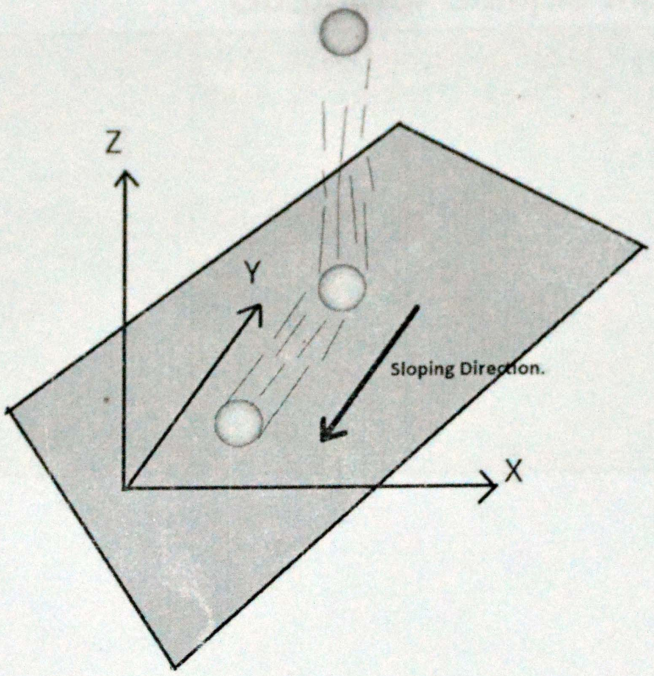
```
Case 1: No match
Case 2: Matching - Monochromatic
Case 3: Matching - Analogous
Case 4: Matching - Complementary
```

Problem G

Input: Standard Input
Output: Standard Output

Gomo's Landing

Toro is the chief controlling officer of spaceship Gomo. Gomo is a uniform sphere of radius **R**.



He needs to land the spaceship on a runway. Due to some architectural reason the runway was not created parallel to the **XY** plane. It creates **t** (**0° < t ≤ 45°**) angle with the **XY** plane. At the time of landing Gomo goes vertically straight down (towards the direction of negative **Z** axis) towards the runway. Just after touching the runway it rolls **D** meters down, i.e. towards the sloping direction to complete landing. Sloping direction of a sloping plane is the direction to which a sphere rolls down due to gravity when it is put on that plane. For this problem sloping plane is a plane which creates more than **0°** angle with **XY** plane. Toro needs to calculate the final position (position of center) of Gomo after landing is completed .

For this problem consider the runway as an infinite plane. It is confirmed that Gomo is more than **R** meter away from the runway. That means, the center of Gomo is more than **2R** meter away.

Input

The first line of the input contains a positive integer **T** (**T ≤ 50**) indicating the number of test cases followed by a blank line.

Each set of input of **T** test cases consists of five lines. First line contains two integers **D** (**0 ≤ D ≤ 1000**) and **R** (**0 ≤ R ≤ 100**). Second line contains three integer **X**, **Y** and **Z** indicating the coordinates of the center of Gomo just before the landing process was started. Each of the remaining three lines contains three integers each,

representing the **X**, **Y** and **Z** coordinates of three *noncollinear* random points of the runway. **The absolute values of all the coordinates are not greater than 1000.**

Consecutive test cases are separated by a single blank line.

Output

For each case, print the case number first, then print **X**, **Y** and **Z** coordinates of Gomo's center after ending the landing process separated by a single space. Errors less than 10^{-3} will be ignored.

Sample Input

```
2

100 10
0 0 100
0 10 0
10 0 10
-10 0 10

100 10
0 0 100
0 10 0
10 0 10
-10 0 8
```

Output for Sample Input

```
Case 1: 0.00000 70.71068 -46.56854
Case 2: -8.18573 73.67156 -44.63224
```

Problem H

Input: Standard Input
Output: Standard Output

Rounding

You will be given an arbitrary large floating-point number and you will have round it to the nearest **t** digits after the decimal point. For example if you round 23.4567, up to three digits after the decimal point you will get 23.457, but if you round it up to six digits after the decimal point you will get 23.456700.

Input

First line of the input file contains an integer **N** (**N ≤ 100**) which denotes the number of test cases. Each of the next **N** lines contains a positive floating-point number **F** (Can have at most **100** digits and must have a decimal point) followed by a positive integer **t** (**t ≤ 50**) which denotes that the number should be rounded upto **t** digits after the decimal point. For simplicity all the digits in the floating-point number should be less than **9**. **But the number may have some leading zeros**.

Output

For each line of input produce one line of output which denotes the number rounded upto **t** digits after the decimal point. **There must be at least one digit before and after the decimal point but there should not be any leading zeros**.

Sample Input

```
3
2.22222 3
00.283928382 3
2.22 5
```

Output for Sample Input

```
2.222
0.284
2.22000
```

Rules for Rounding

Here's the general rule for rounding:

- **If the number you are rounding is followed by 5, 6, 7, 8, or 9, round the number up.** Example: 38 rounded to the nearest ten is 40
- **If the number you are rounding is followed by 0, 1, 2, 3, or 4, round the number down.** Example: 33 rounded to the nearest ten is 30
- 7.8199 rounded to one digit after the decimal point is 7.8
- 1.0621 rounded to two digits after the decimal point is 1.06
- 3.8785 rounded to three digits after the decimal point is 3.879

Problem I

Input: Standard Input
Output: Standard Output

Strange Island

You already have heard the name of Captain Pirate Jack Dumb. He lives in Black iceland, There are something you need to know about Black iceland. There are **N** cities in the iceland. All the cities are numbered from **1** to **N**. The adjacent of city **i** are city **i-1** and **i+1**. Every city except the first and last one has two adjacents. To move to any adjacent city it takes 1 unit time. Any number of people can move in either direction at any time. Amongst the **N** cities there are **M** cities where people lives in. These are denoted as C_1, C_2, ..., C_m.

All the roads, bridge and culverts of the black iceland are made of different types of bamboo woods. Rainy Season is coming. Monsoon rain is expected to destroy or damage those bamboo-wooden roads, bridge and culverts.

Government of Black Iceland believe in FWLW (Firstly word lastly work). They are going to declare **K** different cities from **1** to **N** as mega cities today and then they will do infrastructural development. People living in different cities will have to move to any of those mega city. The government want to set up those **K** megacities such that for all people the maximum amount of time to move to their nearest megacity is minimized.

For example: Let, **N=5, M=2, C_1=1, C_2=5 and K=1**, It will take 2 units time for everyone to move to any of the megacity (Declare City-3 as megacity).
Another example: Let, **N=5, M=2, C_1=1, C_2=4 and K=1**, It will take 2 unit time for everyone to move to any of the megacity (Declare City-2 or City-3 as megacity).

Input

Input starts with an integer **T (1 ≤ T ≤ 150)** denoting the number of test cases. First line of each test cases starts with three integer **N (1 ≤ N ≤ 10^9)**, **M (1 ≤ M ≤ 10000)** and **K (1 ≤ K ≤ N)**. Next line of each test case contains **M** space separated integers C_1, C_2,..., C_m denoting the cities where people lives in.

Output

For each test case print the case number followed by the shortest amount of time required.

Sample Input	Output for Sample Input
3 5 2 1 1 5 5 2 1 1 4 100 6 3 1 23 56 60 79 100	Case 1: 2 Case 2: 2 Case 3: 11

Problem J

Input: Standard Input
Output: Standard Output

To Move Or Not To Move

Consider a rectangle, and a convex polygon determined by the convex hull of a set of points in 2d plane inside that rectangle. You are allowed to do two types of operations on the polygon:

1. Translate the entire polygon by any distance, in any direction. The cost for this operation is equal to the distance. In other words, if before translation any point in the polygon is (x_1, y_1) and after translation the same point is (x_2, y_2), then the cost is equal to the euclidean distance between these two points.
2. Rotate the polygon with respect to a fixed point. The fixed point must be calculated in a way such that the maximum distance of any point of the polygon from that fixed point is minimum. This operation is free of cost.

You have to execute set of operations on the polygon so that the intersection area of the convex polygon and the rectangle is zero. You need to **minimize** the total cost.

Carefully note that:

- You will be given a set of points, and the convex polygon of our interest is determined by the convex hull of that set of points. Each of the points is inside or on the boundary of the rectangle.
- The rectangle is not necessarily axis parallel, hence any three corner points will be given without any particular order.
- In Euclidean geometry, a translation is a geometric transformation that moves every point of a figure or a space by the same distance in a given direction.
- The convex hull of a set of points is the smallest convex polygon that contains all the points of it.
- All points of the polygon will be unique and the area of the polygon will always be positive.

Input

The first line will contain a single positive integer **T** (**T ≤ 100**), the number of test cases.

Of each test cases, the first line will contain six space separated integers. These integers denote (x, y) coordinates of three points of the rectangle.

Next line will contain a positive integer **N** (**3 ≤ N ≤ 50**), denoting the number of points of the set. Each of the next **N** lines will contain two space separated integers denoting (x, y) coordinates of a point.
Absolute value of any coordinate **≤ 10^4**

Output

For each test case, print the case number and the total cost of operations required. Error less than **10^{-3}** will be ignored. **See Sample I/O for details**.

Sample Input

```
3
0 0 20 0 20 20
10
0 0
10 0
0 10
10 10
0 5
5 0
10 5
5 10
2 3
9 7
-2 -2 18 -2 18 18
10
0 0
10 0
0 10
10 10
0 5
5 0
10 5
5 10
2 3
9 7
0 0 20 1 -1 20
10
0 0
10 1
0 10
10 10
0 5
5 1
10 5
5 10
2 3
9 7
```

Output for Sample Input

```
Case 1: 10.00000
Case 2: 12.00000
Case 3: 9.22174
```