Official Problem Set

DO NOT OPEN UNTIL CONTEST BEGINS

2017 ACM ICPC ASIA Kharagpur REGIONAL CONTEST

Problem code: **SCIENCEF**
Problem name: **Science Fair**

This is a story of city Chef Land. The transport network of the city contains **V** intersections and **E** bidirectional roads. The intersections are numbered from 1 to **V**. Every intersection can be reached from every other intersection. The road $\mathbf{E_i}$ has length $\mathbf{W_i}$. The Mayor is organizing a **Science Festival** at intersection **F**.

The **N** students studying in School of Chef Land have requested their principal to send them to the Festival. Each student lives at an intersection (student **i** lives at intersection $\mathbf{P_i}$). The school is present at intersection **S**.

The principle wants to send randomly some of the **N** students to the Science Festival. He writes their names on an Official List and gives it to the bus driver to arrange the transport. The probability that name of $\mathbf{i^{th}}$ student is written on the list is equal to the percentage marks $\mathbf{M_i}$ he receives in Science.

The driver's job is to drive the school bus from school **S** to Festival **F** picking up all the students mentioned on the official list. Since all the students are super excited to visit the festival, therefore if the bus goes through the intersection they are living on, they will on board the bus even if their name was not written on the official list by the principal.

We know that the bus driver doesn't like talkative people in the bus. Also, for every student **i** we know his talkativeness value $\mathbf{T_i}$. The talkativeness of the trip done by driver from **S** to **F** is defined as the **product** of Talkativeness of all the students that got into the bus.

For a trip: The driver faces a **cost(Trip)** = **Length(Trip)** + **(Talkativeness(Trip) modulo 10^9+7)**. Therefore the driver always chooses the path with minimum cost. He also charges this amount from the principal.

If the driver receives an **empty list** he cancels the trip and the cost is **0.**

The principal wants to know the expected value of money paid by him to the driver for a single trip.

Input

- The first line contains 4 space separated integers **V, E, S, F**.
- The second line contains a single integer **N**.
- **i^{th}** line of the next **N** lines contains 3 integers each **P_i T_i M_i.**
- The next **E** lines contains 3 integers each **u v w** denoting a road of length **w** between intersections **u** and **v**

Output

Print a single integer **$(A*B^{-1}) \bmod 10^9+7$** where **A/B** is the expected money paid by the principal and gcd(A, B) = 1. **B^{-1}** is the modular inverse of **B** modulo **10^9+7**

Constraints

- **$1 \leq V \leq 1000$**
- **$1 \leq E \leq 3000$**
- **$1 \leq N \leq 16$**
- **$1 \leq S, F, P_i \leq V$**
- **S, F, P_i**'s are all distinct
- **$0 \leq M_i \leq 100$**
- **$0 \leq T_i \leq 10^9+6$**
- **$1 \leq w \leq 10^6$**
- **$1 \leq u, v \leq V$**
- There is at most 1 road between any pair of intersection
- Every intersection can be reached from every other intersection.
- **$u \neq v$**

Example

Input:

```
5 5 1 5
3
2 1 50
3 1 50
4 1 50
1 2 1
1 3 1
1 4 1
2 5 1
3 5 1
```

Output:

```
125000005
```

Explanation

Official List	Probability	Path travelled	Length	Students in trip	Talk Value	Cost
{}	1/8	None	0	None	0	0
{2}	1/8	1-2-5	2	{2}	1	3
{3}	1/8	1-3-5	2	{3}	1	3
{4}	1/8	1-4-1-3-5	4	{3, 4}	1*1	5
{2, 3}	1/8	1-2-1-3-5	4	{2, 3}	1*1	5
{2, 4}	1/8	1-4-1-2-5	4	{2, 4}	1*1	5
{3, 4}	1/8	1-4-1-3-5	4	{3, 4}	1*1	5
{2, 3, 4}	1/8	1-4-1-2-1-3-5	6	{2, 3, 4}	1*1*1	7

Expected cost = (0+3+3+5+5+5+5+7)/8 = 33/8 = $33 * 8^{-1}$ **mod 10^9+7**= 33*125000001 **mod 10^9+7 =** 125000005

Problem code: **DISCAREA**
Problem name: **Black Discs**

Consider the 2-d plane which is white. You are given **n** semi-discs on it which are colored black. They are placed on the x - axis at various positions, in such a way that its center is on the x-axis and the entire semi-disc is **on or above** the x-axis. By center of a semi-disc, we mean the center of the disc of which it is a part. For each semi-disc you know its radius and the coordinate of its center in the x-axis. There may be overlaps or one may be completely inside / outside another semi-disc.

There are **q** queries. In each query you will be given a circle with radius **r** and center C at coordinates (x, y). It is guaranteed that $y \geq r$ (meaning that no part of the circle is below the x-axis). You need to find the area within the circle *which is **black***. Meaning, area of the circle which is overlapping with some part of any semi-disc.

Input

- The first line contains a single integer, **T**, which denotes the number of testcases. The description of each testcase follows.
- The first line of each testcase contains two space-separated integers: **n** and **q**, denoting the number of semi-discs and the number of queries respectively.
- The i-th of the next **n** lines contains two space-separated integers: $\mathbf{x_i}$ and $\mathbf{r_i}$, denoting that the i-th semi-disc has its center at ($\mathbf{x_i}$, 0) and has radius $\mathbf{r_i}$.
- The i-th of the next **q** lines contains three space-separated integers: $\mathbf{x_i}$, $\mathbf{y_i}$ and $\mathbf{r_i}$, denoting that the i-th query circle has its center at ($\mathbf{x_i}$, $\mathbf{y_i}$) and has radius $\mathbf{r_i}$.

Output

- For each query in a testcase, output a single line with the answer.
- Your output will be considered correct if both these are true:
    - In each query, the absolute error is less than or equal to 0.02.
    - In a single testcase, the average of absolute error over all queries must be less than 0.01.

Constraints

- $1 \leq \mathbf{T} \leq 10$
- $1 \leq \mathbf{n} \leq 100000$
- $20 \leq \mathbf{q} \leq 200$
- $1 \leq$ all x and y coordinates in the input ≤ 100000
- $1 \leq$ all radii in input ≤ 100000
- Sum of **n** over all testcases ≤ 100000
- Sum of **q** over all testcases ≤ 200

Example

Input:

```
1
4 2
1 2
2 3
4 2
7 3
7 2 2
8 3 3
```

Output:

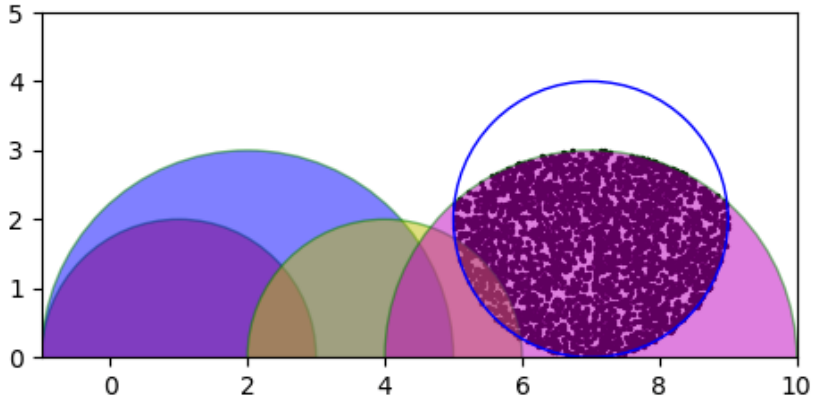
```
9.3204778956
10.2198951180
```

Explanation

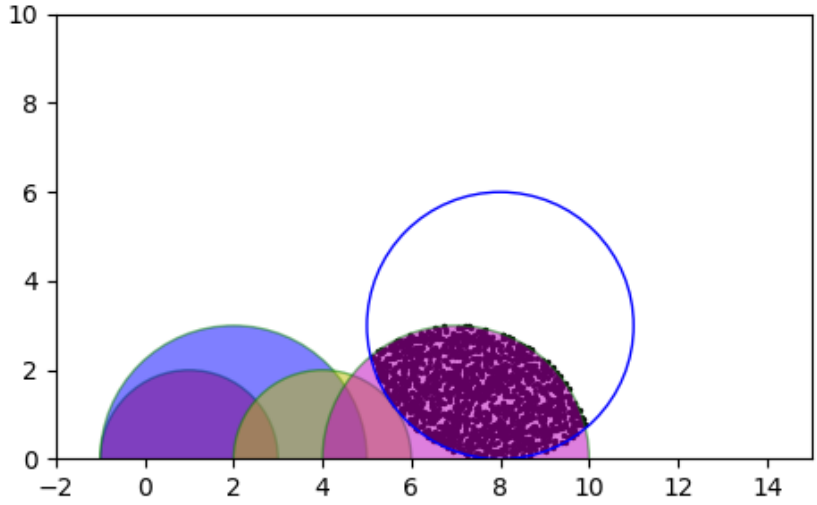
Note that the example shown above has only two queries (for brief representation) and hence doesn't satisfy the $20 \leq \mathbf{q}$ constraint. However, in the original test data the constraints will be followed.

The black semi-discs have been colored with various colors for better understanding. And the dotted region represents the region whose area we want to find.

Query 1: The figure below represents the scenario:



Query 2: The figure below represents the scenario:



Problem code: **STRLBP**
Problem name: **Uniform Strings**

You are given a string **s** of length 8 consisting solely of '0's and '1's. Assume that the characters of the string are written in a circular fashion. You need to find the number of 0-1 or 1-0 transitions that one has to make while making a single traversal over the string. ie. start from any character and go circularly until you get back to the same character, and find the number of transitions that you made. The string is said to be said to be *uniform* if there are at most two such transitions. Otherwise, it is called *non-uniform*.

Given the string **s**, tell whether the string is *uniform* or not.

Input

The first line of the input contains an integer **T** denoting the number of test cases. The description of **T** test cases follows.

The only line of input contains the string **s**.

Output

For each test case, output "uniform" if the given string is *uniform* and "non-uniform" otherwise.

Constraints

- **1 ≤ T ≤ 256**
- Length of **s** is 8

Example

Input

```
4
00000000
10101010
10000001
10010011
```

Output

```
uniform
non-uniform
uniform
non-uniform
```

Explanation

The number of transitions are 0, 8, 2 and 4 for the respective cases. So, the first and third one are uniform while the second and fourth one are non-uniform.

Problem code: **SADQ**
Problem name: **SAD Queries**

"The Mean Absolute Difference is a measure of statistical dispersion equal to the average absolute difference of two independent values drawn from a probability distribution." - Wikipedia

In this problem we are concerned not with Mean Absolute Difference (MAD) of two distributions but rather with the Summed Absolute Difference (SAD) of two arrays. Given arrays (1-indexed) **P** and **Q** of lengths **p** and **q**, define

$$SAD(P,Q) = \sum_{i=1}^{p} \sum_{j=1}^{q} |P_i - Q_j|$$

Given a collection of **K** arrays $\mathbf{A_1}$, ..., $\mathbf{A_K}$, report SAD($\mathbf{A_i}$, $\mathbf{A_j}$) for several queries (**i**, **j**).

Input

- The first line of input contains a single positive integer **K**.
- Each of the next **K** lines starts with a positive integer $\mathbf{s_i}$, the size of $\mathbf{A_i}$, followed by $\mathbf{s_i}$ space-separated integers $\mathbf{A_{i,1}}$, ..., $\mathbf{A_{i,si}}$ which denotes the array $\mathbf{A_i}$.
- The next line of input consists of a single positive integer **M**.
- Each of the next **M** lines consists of 2 positive integers **i** and **j** which lie between 1 and **K** (inclusive), specifying the indices of the arrays for which the SAD value must be reported.

Output

Output **M** lines. Line **i** should contain the answer to query **i**.

Constraints

- $1 \leq \mathbf{K}, \mathbf{M} \leq 5 * 10^5$
- $1 \leq \mathbf{s_1} + ... + \mathbf{s_K} \leq 5 * 10^5$
- $-10^7 \leq \mathbf{A_{i,j}} \leq 10^7$
- $1 \leq \mathbf{i}, \mathbf{j} \leq \mathbf{K}$ in each of the queries

Example

Input:

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

Output:

```
30
30
60
```

Explanation:

Query 1: The arrays in question are $\mathbf{A_3}$ = [2, 4] and $\mathbf{A_4}$ = [4, -3, -4, 4].

SAD($\mathbf{A_3}$, $\mathbf{A_4}$)

= |2 - 4| + |2-(-3)| + |2 -(-4)| + |2 - 4| + |4 - 4| + |4 - (-3)| + |4 - (-4)| + |4 - 4|

= 2 + 5 + 6 + 2 + 0 + 7 + 8 + 0

= 30

Problem code: **XORQUERY**
Problem name: **Chef and XOR Queries**

Mr. X has given the Chef an undirected tree T with **n** nodes numbered from 1 to **n**. Each edge $\mathbf{e}_i$ of the tree has a non-negative integer $\mathbf{w}_i$ written on it. But the edge weights are hidden from you. You have access only to the structure of the tree. ie. you know all the edges $\mathbf{e}_i$.

Mr. X. has taught Chef how to compute the function f(**u**, **v**) = Bitwise *XOR* of all the numbers present on the edges in the unique path from **u** to **v**.

He now wants to test his disciple's understanding of this funciton. He tells the Chef various values of this function one by one. Since Mr. X. wants to test Chef's skills, this information could be wrong sometimes. You being good friends with the Chef have been asked to help him out to pass the test.

Total **Q** events occur each of which can be in one of the following forms:

- **1 u v r**:
  Mr. X says that f(**u**, **v**) = **r**. For this type of event you must print:
  - *AC* if the given information is consistent with all the previous information that you have accepted as AC and there is *no way* you can argue that this information is wrong. Accept this new information as correct.
  - *WA* when you can prove that the given information is wrong and does not fit in with the previously *AC* events.
- **2 u v**:
  Print the value of f(**u**, **v**) using only the information provided in the previously *AC* events.
  If it is not possible to correctly determine this value based on only previously *AC* events print -1.

Input

- The first line of the input contains an integer **T** denoting the number of test cases. The description of **T** test cases follows.
- First line of each test case contains two space separated integers **n** and **Q** denoting the number of nodes and number of events respectively.
- Next **n** - 1 lines each contain two space separated integers **u v** denoting that there is an edge between **u** and **v** in the tree.
- Description of **Q** events follow. Each line follows one of these two formats:
  - **1 u v r**, denoting first type of event.
  - **2 u v**, denoting second type of event.

Output

For each event print the answer to the event as described.

Constraints

- **1 ≤ T ≤ 2000**
- **1 ≤ n ≤ 10^5**
- **1 ≤ Q ≤ 2 * 10^5**
- **1 ≤ u, v ≤ n**
- **1** ≤ sum of **n** over all the testcases ≤ **10^5**
- **1** ≤ sum of **Q** over all the testcases ≤ **2 * 10^5**
- **0 ≤ r ≤ 2 * 10^9**
- **0 ≤ w_i**
- The graph represented by **e_i** is a tree

Example

Input

```
1
4 6
1 2
2 3
3 4
1 1 1 10
1 1 2 2
1 2 3 4
1 1 3 7
2 1 3
2 3 4
```

Output

```
WA
AC
AC
WA
6
-1
```

Explanation

- Event 1: f(1, 1) cannot be 10. Since there is no edge between 1 and 1, it must be 0 ⇒ *WA*
- Event 2: You cannot argue that f(1, 2) cannot be 2. Therefore you accept it ⇒ *AC*. Thus now we know that the weight of the edge between 1 and 2 is 2.
- Event 3: You cannot argue that f(2, 3) cannot be 4. Therefore you accept it ⇒ *AC*. Thus now we know that the weight of the edge between 2 and 3 is 4.
- Event 4: f(1, 3) is xor of weights of edges 1-2 and 2-3 = 2^4 = 6. Given information is wrong ⇒ *WA*
- Event 5: f(1, 3) is 6 as calculated above ⇒ 6
- Event 6: f(3, 4) cannot be answered using only the above information ⇒ -1

Problem code: **TAXITURN**
Problem name: **Taxi Making Sharp Turns**

Taxis of Kharagpur are famous for making sharp turns. You are given the coordinates where a particular taxi was on a 2-D planes at **N** different moments: $(\mathbf{x_1}, \mathbf{y_1})$, $(\mathbf{x_2}, \mathbf{y_2})$, ..., $(\mathbf{x_N}, \mathbf{y_N})$. In between these coordinates, the taxi moves on a straight line. A turn at the i-th ($2 \leq i \leq \mathbf{N}$-1) coordinate is said to be a *sharp* turn if the angle by which it turns at Point B = $(\mathbf{x_i}, \mathbf{y_i})$ when going from coordinates A = $(\mathbf{x_{i-1}}, \mathbf{y_{i-1}})$ to C = $(\mathbf{x_{i+1}}, \mathbf{y_{i+1}})$ via $(\mathbf{x_i}, \mathbf{y_i})$ is greater than 45 degrees. ie. suppose you extend the line segment AB further till a point D, then the angle DBC would be greater than 45 degrees.

You have to identify whether the taxi made a sharp turn somewhere or not. If it made a sharp turn, also identify whether it is possible to change the coordinates at one of the **N** moments to make sure that the taxi doesn't make any sharp turn. Note that all the **N** pairs of coordinates (including the new coordinates) should be distinct and should have their x and y coordinates at least 0 and at most 50.

Input

- The first line of the input contains a single integer **T** denoting the number of test cases. The description of **T** test cases follows.
- The first line of each test case contains a single integer **N** denoting the number of moments at which you are given the information of the taxi's coordinates.
- Each of the next **N** lines contains two space-separated integers $\mathbf{x_i}$ and $\mathbf{y_i}$ denoting the x and y coordinates of the taxi at i-th moment.

Output

For each test case, print a single line containing two space separated strings, either of which can be a "yes" or "no" (without quotes). First you should tell whether the taxi made a sharp turn or not. Second you should tell whether it is possible to modify at most one coordinate in such a way that taxi doesn't make a sharp turn. Note that if the first string is "yes", then the second string would always be "yes".

Constraints

- $1 \leq \mathbf{T} \leq 50$
- $3 \leq \mathbf{N} \leq 50$
- $0 \leq \mathbf{x_i}, \mathbf{y_i} \leq \mathbf{50}$
- It's guaranteed that all ($\mathbf{x_i}, \mathbf{y_i}$) pairs are distinct.

Example

Input

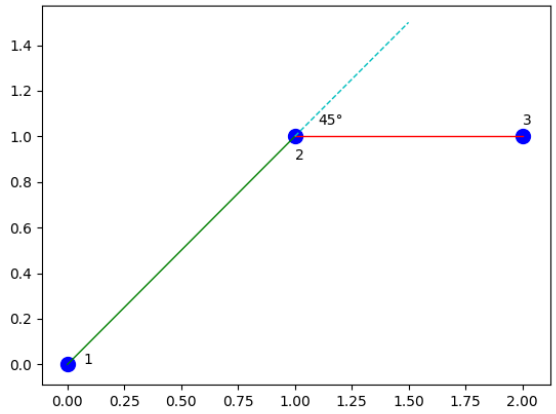
```
5
3
0 0
1 1
2 1
3
0 0
1 0
6 1
3
0 0
1 0
1 1
4
0 0
1 0
1 1
6 1
6
0 0
1 0
1 1
2 1
2 2
3 2
```

Output

```
yes yes
yes yes
no yes
no yes
no no
```

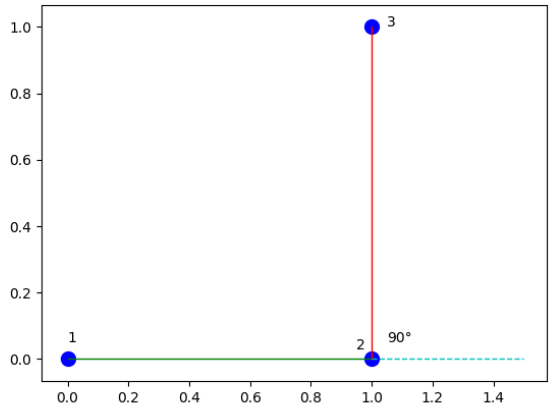
Explanation

Example 1.



You can see that taxi is never making a sharp turn.

Example 3



You can see that taxi is making a sharp turn of 90 degrees, an angle greater than 45'. However, you can change the coordinates of the third points to (2, 1) to ensure that the angle remains <= 45'.

Problem code: **SPAMCLAS**
Problem name: **Spam Classification Using Neural Net**

Neural nets are extremely popular in the Machine Learning domain. A neural net is composed of multiple layers. It has an input layer in which you input the parameter **x** (the input of the program). The input is then passed through multiple hidden layers, finally getting one output at the final layer, called the output layer.

We have a very simple neural net, which consist of **N** hidden layers. Each layer contains one neuron. Each neuron has two values associated with it: $\mathbf{w_i}$, and $\mathbf{b_i}$, denoting the weight and the bias of the neuron. If you give the neuron an input of x, it produces an output of $(\mathbf{w_i} * \mathbf{x}) + \mathbf{b_i}$.

Thus, an input **x** gets transformed by the neural net as follows. The first hidden neuron takes the input **x** and produces $\mathbf{y} = \mathbf{w_1} * \mathbf{x} + \mathbf{b_1}$, which acts as the input for the second neuron. Then, the second neuron takes input **y** and produces an output of $\mathbf{z} = \mathbf{w_2} * \mathbf{y} + \mathbf{b_2}$. This keeps happening and you get a single output at the end from the **N**-th neuron.

There are some users and we want to find if they are spamming or not. They have integer user-ids, which range from **minX** to **maxX** (both inclusive). So we take each of these user-ids and feed it as input to the first layer of the neural net. If the final output is even, then that user is not a spammer, otherwise, the user is a spammer. You have to count the number of non-spammers and spammers.

Input

- The first line of the input contains a single integer **T** denoting the number of test cases. The description of **T** test cases follows.
- The first line of each test case contains three space-separated integers **N, minX, maxX**.
- Each of the next **N** lines contains two space-separated integers $\mathbf{w_i}$ and $\mathbf{b_i}$ denoting the weight and the bias of the i-th neuron.

Output

For each test case, output two space-separated integers denoting the number of non-spammers and the number of spammers, respectively.

Constraints

- $1 \le \mathbf{T} \le 10$
- $1 \le \mathbf{N} \le 10^5$
- $1 \le \mathbf{minX} \le \mathbf{maxX} \le 10^9$
- $1 \le \mathbf{w_i}, \mathbf{b_i} \le 10^9$

Example

Input

```
3
1 1 2
1 2
2 1 4
2 4
2 3
3 2 1000000000
2 4
2 2
5 4
```

Output

```
1 1
0 4
999999999 0
```

Explanation

Example 1. There is a single neuron with weight = 1 and bias = 2. Let us check the output for x = 1: w * x + b = 1 * 1 + 2 = 3. Output for x = 2 would be 2 * 1 + 2 = 4. You can see that one of these is even and the other is odd. So, there is one spammer and one non-spammer.

Example 2. There are two neurons with weights 2 each, but bias 4, 3 respectively.

- x = 1, y = 2 * 1 + 4 = 6. z = 2 * 6 + 3 = 15
- x = 2, y = 2 * 2 + 4 = 8. z = 2 * 8 + 3 = 19
- x = 3, y = 2 * 3 + 4 = 10. z = 2 * 10 + 3 = 23
- x = 4, y = 2 * 4 + 4 = 12. z = 2 * 12 + 3 = 27

You can see that all of these are odd and hence signify that they are spammers. So, there are 0 non-spammers and 4 spammers.

Problem code: **NONOVSEG**
Problem name: **Non Overlapping Segments**

You are given **N** horizontal segments each of length **R**. You are given the x-coordinates of their left end-points: the i-th segment has the left end point at $\mathbf{x_i}$. You can see that the right end point will be at $\mathbf{x_i + R}$.

You want to move the segments such that they lie entirely in the range **[0, L]**. After the movements, no two segments should overlap, however, they are allowed to touch each other.

Find out the minimum number of segments that you need to move to achieve this. In other words, you want to maximize the number of segments that are left untouched. It's guaranteed that the sum of the lengths of the segments is such that they all can fit in **[0, L]**.

Input

- The first line of the input contains a single integer **T** denoting the number of test cases. The description of **T** test cases follows.
- The first line of each test case contains three space-separated integers **N, L, R**.
- The second line of each test case contains **N** space-separated integers. The i-th integer denotes the coordinate of the left end point of i-th segment, i.e. $\mathbf{x_i}$.

Output

For each test case, output a single integer corresponding to the minimum number of segments that you need to move.

Constraints

- $1 \leq \mathbf{T} \leq 2500$
- $1 \leq \mathbf{N} \leq 500$
- $1 \leq \mathbf{L, R} \leq 10^9$
- $-10^9 \leq \mathbf{x_i} \leq 10^9$
- Sum of **N** over all test cases in a single test file won't exceed 2500.
- $\mathbf{L} \geq \mathbf{R} * \mathbf{N}$, i.e. all the segments can fit inside the range.

Example

Input

```
3
4 4 1
0 1 2 3
4 4 1
-1 2 3 5
4 4 1
2 2 2 2
```

Output

```
0
2
3
```

Explanation

Example 1. All the segments lie in the range [0, **L**]. None of the segments overlap with each other. Hence, we don't need to move any segment at all.

Example 2. You can move the first segment to [0, 1]. You can also move the fourth segment to [1, 2]. After this movement, you can see that the all the segments lie in the range [0, **L**] and none of the segments overlap with each other. Overall, we moved 2 segments. There is no way to achieve this by moving less than 2 segments.

Problem code: **SPANTREE**
Problem name: **Spanning Tree**

You are given an undirected weighted connected graph with **N** vertices numbered from 1 to **N**. Chef knows all the edges of the graph but will only reveal information if you give him some money.

You have $\mathbf{10^4}$ **coins**. You are allowed to ask the chef a certain type of query multiple times. In a single query, you will provide **2** non-intersecting, non-empty set of vertices. Let **A** be the first set and **B** be the second set. Chef then will respond by providing the minimum (least) weight edge between any pair of nodes **u** and **v**, such that node **u** is in set **A** and node **v** is in set **B**. In particular, he will return the end points of the least weight edge along with the weight of this edge (in case of a tie, Chef will choose arbitrarily). If there is no such edge, Chef returns **-1**. A single query costs you **|A|** coins, where |X| denotes the size of the set X. Do note that the Chef does NOT like too many numbers, so he has allowed the sum of **|A| + |B|** over all the queries to be at most $\mathbf{2*10^6}$

Your aim is to determine the sum of all the weights in a minimum spanning tree of this graph using at most 10^4 coins.

Input and Output

You can interact with the judge using the standard input and output.

The first line of the input contains a single integer **N**, denoting the number of vertices.

You need to print to the standard output, for the operations you wish to perform. There are two types of operations:

- In the first type, you are asking the judge the least weight edge between 2 sets, **A** and **B**. You need to print **3** lines.
  - The first line starts with **1 |A| |B|**, where **|A|** and **|B|** are the size of the 2 sets.
  - The second line contains **|A|** integers, denoting the elements of the set **A**.
  - The thirdline contains **|B|** integers, denoting the elements of the set **B**.

The judge will return a triplet of integers by printing the integers in the standard input. It will be in the format **u v w**, where **u** and **v** denote the end points of the least weight edge between the 2 sets and **w** denotes the cost of this edge. In case there is no such edge then **u = v = w = -1**

- In the second type, you are telling the judge the answer. For each test case you should perform this operation exactly once at the end. It does not cost you any coins.

  You will print the answer to the standard output in the format **2 X**,
  where **X** should be the sum of all edge weights in a minimum spanning tree of this graph.

Note

Don't forget to flush the standard output after printing each line. It can be done using fflush(stdout) in C/C++, System.out.flush() in Java and sys.stdout.flush() in Python.

If the **2** sets **A** and **B** given in the input are intersecting OR one of them is empty OR there are duplicate elements in one of them OR one of them has an element which does NOT belong to the range **[1, N]**, then your program will get the verdict Wrong Answer.

Also if you cross the limit of 10^4 coins OR the sum of $|A| + |B|$ over all the operations of type 1 exceeds the limit of $2*10^6$, then your program will get the verdict Wrong Answer.

The query of second type don't cost you any coin.

Constraints

- $2 \le N \le 10^3$

  For each query -

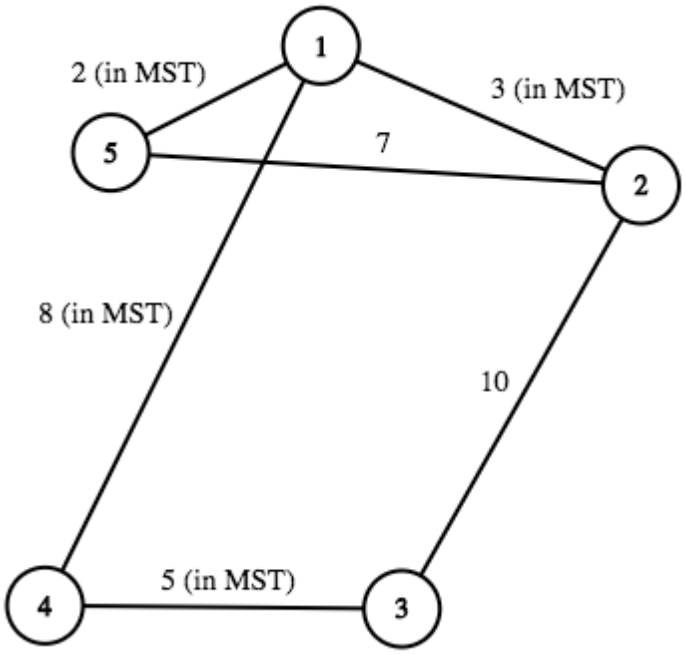
- $1 \le u, v \le N$
- $1 \le w \le 10^5$

Example

Input / Judge Feedback	Your output / Your query
5	
	1 1 1 1 5
1 5 2	
	1 1 2 2 1 4
2 1 3	
	1 2 2 1 2 4 3
1 4 8	
	1 1 1 4 5
-1 -1 -1	
	1 1 2 2 3 5
2 5 7	
	1 3 1 4 2 5 3
4 3 5	
	2 18

Explanation

The graph (which is hidden) in the sample has 5 nodes and the sum of the weights in the MST is 18.



- The cost of the first query is 1 coin.
- The cost of the second query is 1 coin.
- The cost of the third query is 2 coins.
- The cost of the fourth query is 1 coin.
- The cost of the fifth query is 1 coin.
- The cost of the sixth (last) query is 3 coins.

So the total amount of coins used = 1 + 1 + 2 + 1 + 1 + 3 = 9 coins which is ≤ 10,000 coins

Note - The above queries are not an ensured method of finding the correct answer given the graph had different edges, so please do NOT try to make any logical deductions using the sample queries.

Problem code: **GENPERM**
Problem name: **Generating A Permutation**

For a permutation $P = (p_1, p_2, ..., p_N)$ of numbers [1, 2, ..., **N**], we define the function $f(P) = \max(p_1, p_2) + \max(p_2, p_3) + ... + \max(p_{N-1}, p_N)$.

You are given **N** and an integer **K**. Find and report a permutation P of [1, 2, ..., **N**] such that f(P) = **K**, if such a permutation exists.

Input

- The first line of the input contains an integer **T** denoting the number of test cases. The description of **T** test cases follows.
- The only line of each test case consists of two space-separated integers **N, K** respectively.

Output

For each test case, if a permutation satisfying the condition exists, output a single line containing **N** space-separated integers which denotes any such permutation. If no such permutation exists, output a single integer -1 instead.

Use fast I/O methods since the size of the output is large.

Constraints

- $1 \leq \mathbf{T} \leq 40$
- $1 \leq \mathbf{N} \leq 10^5$
- Sum of **N** over all test cases in each file $\leq 10^6$
- $0 \leq \mathbf{N} \leq 2 * 10^{10}$

Example

Input:

```
3
4 12
2 2
5 14
```

Output:

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

Explanation

Example 1. There doesn't exist any permutation of numbers [1, 2, 3, 4] that can have its f value equal to 4. Hence answer is -1.

Example 2. The permutations [1, 2] and [2, 1] both have their f values equal to 2. You can print any of these two permutations.

Example 3. The permutation [5, 4, 3, 2, 1] has f value = max(5, 4) + max(4, 3) + max(3, 2) + max(2, 1) = 5 + 4 + 3 + 2 = 14.

Problem code: **NUMBGAME**
Problem name: **Number Game**

You are playing a game with a robot. The game starts with two integers: **A** and **M**. The robot makes exactly one move in the entire game, and it does so at the very beginning - it will remove exactly 1 digit from **A** and output it as the starting value (say *X*). Note that the value **A** remains intact. It is not changed by the robot. After the robot makes its moves, it is your turn. You can make an unlimited number of moves. In each move, you must remove exactly 1 digit from A and append it to *X* (to the right side of X). Again note that none of your moves change the value of **A**. You win if you can eventually make X a multiple of M (i.e. X mod M = 0). How many possible starting moves bot can make such that you are guaranteed to win assuming you play optimally?

Here is an example of a game: Suppose the numbers are **A** = 1003, **M** = 4. The robot can make four possible starting moves: 003 by removing 1st digit, 103 by removing 2nd or 3rd digit (both count as a separate possibilities even though the starting X will be same) or 100 by removing the 4th digit. Let us say the robot chooses 003 as the starting move. Then you can win by appending 100 (by removing the 4th digit) making X = 003100 which is divisible by 4.

Input

- The first line of the input contains an integer **T** denoting the number of test cases. The description of **T** test cases follows
- The only line of each test case contains 2 integers **A** and **M**.

Output

- For each test case, output a single line containing the number of possible first moves that the robot can make such that you can force a win.

Constraints

- **1 ≤ T ≤ 100**
- **10 ≤ A <** 10^{10^6} (i.e. The number of digits in **A** can be upto 10^6)
- **1 ≤ M ≤ 1000**
- The sum of number of digits of **A** over all test cases will not exceed **5 * 10^6**

Example

Input:

```
5
1003 4
11 4
2004 3
123 1
100000 27
```

Output:

```
4
0
4
3
6
```

Explanation

Testcase 1: Whatever the robot's starting move is, you can win by appending 100

Testcase 2: It is not possible to win, whatever the starting move of the robot