A: LongNumber

We generate two infinitely long numbers. The first number is generated by writing down all positive consecutive integers, while the second is generated by writing down all of their squares. We then find the sum of these two numbers. Here are the first 33 digits of the calculation:

```
    12345678910111213141516171819202 1...
+   14916253649648110012114416919622 5...
=   27261932559759323153630588738824 6...
```

The first digit of the result is 2, the second digit is 7, the third is 2 and so on.

Input

There are many test cases. Each test case consist of an integer k ($1 \leq k \leq 2147483647$) on a line by itself.

Output

Output for each test case a line with the digit at position k of the resulting number, where the first digit is at position 1.

Sample Input

```
1
5
78
1000000
1780243932
```

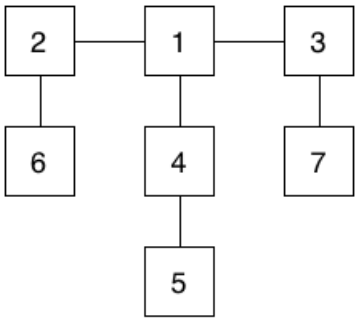
Sample Output

```
2
1
5
6
1
```

B: Balancing Act

Consider a tree T with N ($1 \leq N \leq 20,000$) nodes numbered 1...N. Deleting any node from the tree yields a forest: a collection of one or more trees. Define the balance of a node to be the size of the largest tree in the forest T created by deleting that node from T.

For example, consider the tree:



Deleting node 4 yields two trees whose member nodes are {5} and {1,2,3,6,7}. The larger of these two trees has five nodes, thus the balance of node 4 is five. Deleting node 1 yields a forest of three trees of equal size: {2,6}, {3,7}, and {4,5}. Each of these trees has two nodes, so the balance of node 1 is two.

For each input tree, calculate the node that has the minimum balance. If multiple nodes have equal balance, output the one with the lowest number.

Input

The first line of input contains a single integer t ($1 \leq t \leq 20$), the number of test cases. The first line of each test case contains an integer N ($1 \leq N \leq 20,000$), the number of congruence. The next $N-1$ lines each contains two space-separated node numbers that are the endpoints of an edge in the tree. No edge will be listed twice, and all edges will be listed.

Output

For each test case, print a line containing two integers, the number of the node with minimum balance and the balance of that node.

Sample Input

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

Sample Output

```
1 2
```

C: Crossing the Desert

In this problem, you will compute how much food you need to purchase for a trip across the desert on foot.

At your starting location, you can purchase food at the general store and you can collect an unlimited amount of free water. The desert may contain oases at various locations. At each oasis, you can collect as much water as you like and you can store food for later use, but you cannot purchase any additional food. You can also store food for later use at the starting location. You will be given the coordinates of the starting location, all the oases, and your destination in a two-dimensional coordinate system where the unit distance is one mile.

For each mile that you walk, you must consume one unit of food and one unit of water. Assume that these supplies are consumed continuously, so if you walk for a partial mile you will consume partial units of food and water. You are not able to walk at all unless you have supplies of both food and water. You must consume the supplies while you are walking, not while you are resting at an oasis. Of course, there is a limit to the total amount of food and water that you can carry. This limit is expressed as a carrying capacity in total units. At no time can the sum of the food units and the water units that you are carrying exceed this capacity.

You must decide how much food you need to purchase at the starting location in order to make it to the destination. You need not have any food or water left when you arrive at the destination. Since the general store sells food only in whole units and has only one million food units available, the amount of food you should buy will be an integer greater than zero and less than or equal to one million.

Input

The first line of input in each trial data set contains n ($2 \leq n \leq 20$), which is the total number of significant locations in the desert, followed by an integer that is your total carrying capacity in units of food and water. The next n lines contain pairs of integers that represent the coordinates of the n significant locations. The first significant location is the starting point, where your food supply must be purchased; the last significant location is the destination; and the intervening significant locations (if any) are oases. You need not visit any oasis unless you find it helpful in reaching your destination, and you need not visit the oases in any particular order.

The input is terminated by a pair of zeroes.

Output

For each trial, print the trial number followed by an integer that represents the number of units of food needed for your journey. Use the format shown in the example. If you cannot make it to the destination under the given conditions, print the trial number followed by the word "`Impossible`".

Place a blank line after the output of each test case.

Sample Input

```
4 100
10 -20
-10 5
30 15
15 35
2 100
0 0
100 100
0 0
```

Sample Output

```
Trial 1: 136 units of food

Trial 2: Impossible
```

D: Parking

When shopping on Long Street, Michael usually parks his car at some random location, and then walks to the stores he needs. Can you help Michael choose a place to park which minimises the distance he needs to walk on his shopping round?

Long Street is a straight line, where all positions are integer. You pay for parking in a specific slot, which is an integer position on Long Street. Michael does not want to pay for more than one parking though. He is very strong, and does not mind carrying all the bags around.

Input

The first line of input gives the number of test cases, $1 \leq t \leq 100$. There are two lines for each test case. The first gives the number of stores Michael wants to visit, $1 \leq n \leq 20$, and the second gives their n integer positions on Long Street, $0 \leq x_i \leq 99$.

Output

Output for each test case a line with the minimal distance Michael must walk given optimal parking.

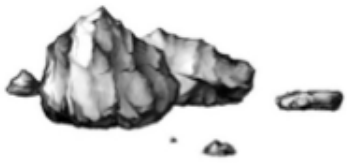
Sample Input

```
2
4
24 13 89 37
6
7 30 41 14 39 42
```

Sample Output

```
152
70
```

E: Easy Climb



Somewhere in the neighborhood we have a very nice mountain that gives a splendid view over the surrounding area. There is one problem though: climbing this mountain is very difficult, because of rather large height differences. To make more people able to climb the mountain and enjoy the view, we would like to make the climb easier.

To do so, we will model the mountain as follows: the mountain consists of n adjacent stacks of stones, and each of the stacks is h_i high. The successive height differences are therefore $h_{i+1} - h_i$ (for $1 \leq i \leq n-1$). We would like all absolute values of these height differences to be smaller than or equal to some number d.

We can do this by increasing or decreasing the height of some of the stacks. The first stack (the starting point) and the last stack (the ending point) should remain at the same height as they are initially. Since adding and removing stones requires a lot of effort, we would like to minimize the total number of added stones plus the total number of removed stones. What is this minimum number?

Input

On the first line one positive number: the number of testcases, at most 100. After that per testcase:

- One line with two integers n ($2 \leq n \leq 100$) and d ($0 \leq d \leq 10^9$): the number of stacks of stones and the maximum allowed height difference.
- One line with n integers h_i ($0 \leq h_i \leq 10^9$): the heights of the stacks.

Output

Per testcase:

- One line with the minimum number of stones that have to be added or removed or "`impossible`" if it is impossible to achieve the goal.

Sample Input

```
3
10 2
4 5 10 6 6 9 4 7 9 8
3 1
6 4 0
4 2
3 0 6 3
```

Sample Output

```
6
impossible
4
```

F: Elevator Trouble

You are on your way to your first job interview as a program tester, and you are already late. The interview is in a skyscraper and you are currently in floor s, where you see an elevator. Upon entering the elevator, you learn that it has only two buttons, marked "UP u" and "DOWN d". You conclude that the UP-button takes the elevator u floors up (if there aren't enough floors, pressing the UP-botton does nothing, or at least so you assume), whereas the DOWN-button takes you d stories down (or none if there aren't enough). Knowing that the interview is at floor g, and that there are only f floors in the building, you quickly decide to write a program that gives you the amount of button pushes you need to perform. If you simply cannot reach the correct floor, your program halts with the message "`use the stairs`".

Given input f, s, g, u and d (floors, start, goal, up, down), find the shortest sequence of button presses you must press in order to get from s to g, given a building of f floors, or output "`use the stairs`" if you can not get from s to g by the given elevator.

Input

There are many test cases. Each test case will consist of one line, namely `f s g u d`,where $1 \leq s, g \leq f \leq 1000000$ and $0 \leq u, d \leq 1000000$. The floors are one-indexed, i.e. if there are 10 stories, s and g be in $[1, 10]$.

Output

For each test case reply with the minimum numbers of pushes you must make in order to get from s to g, or output `use the stairs` if it is impossible given the configuration of the elevator.

Sample Input

```
10 1 10 2 1
100 2 1 1 0
```

Sample Output

```
6
use the stairs
```

G: Race

Disky and Sooma, two of the biggest mega minds of Bangladesh went to a far country. They ate, coded and wandered around, even in their holidays. They passed several months in this way. But everything has an end. A holy person, Munsiji came into their life. Munsiji took them to derby (horse racing). Munsiji enjoyed the race, but as usual Disky and Sooma did their usual task instead of passing some romantic moments. They were thinking in how many ways a race can finish! Who knows, maybe this is their romance!

In a race there are n horses. You have to output the number of ways the race can finish. Note that, more than one horse may get the same position. For example, 2 horses can finish in 3 ways.

1. Both first
2. $horse_1$ first and $horse_2$ second
3. $horse_2$ first and $horse_1$ second

Input

Input starts with an integer T (≤ 1000), denoting the number of test cases. Each case starts with a line containing an integer n ($1 \leq n \leq 1000$).

Output

For each case, print the case number and the number of ways the race can finish. The result can be very large, print the result modulo 10056.

Sample Input

```
3
1
2
3
```

Sample Output

```
Case 1: 1
Case 2: 3
Case 3: 13
```

H: Kingdom

King Kong is the feared but fair ruler of Transylvania. The kingdom consists of two cities and $N < 150$ towns, with nonintersecting roads between some of them. The roads are bidirectional, and it takes the same amount of time to travel them in both directions. Kong has $G < 353535$ soldiers.

Due to increased smuggling of goat cheese between the two cities, Kong has to place his soldiers on some of the roads in such a way that it is impossible to go from one city to the other without passing a soldier. The soldiers must not be placed inside a town, but may be placed on a road, as close as Kong wishes, to any town. Any number of soldiers may be placed on the same road. However, should any of the two cities be attacked by a foreign army, the king must be able to move all his soldiers fast to the attacked city. Help him place the soldiers in such a way that this mobilizing time is minimized.

Note that the soldiers cannot be placed in any of the cities or towns. The cities have ZIP-codes 95050 and 104729, whereas the towns have ZIP-codes from 0 to $N - 1$. There will be at most one road between any given pair of towns or cities.

Input

The input contains several test cases. The first line of each test case is N, G and E, where N and G are as defined above and $E < 5000$ is the number of roads. Then follow E lines, each of which contains three integers: A and B, the ZIP codes of the endpoints, and ϕ, the time required to travel the road, $\phi < 1000$. The last line of the input is a line containing a single 0.

Output

For each test case in the input, print the best mobilizing time possible, with one decimal. If the given number of soldiers is not enough to stop the goat cheese, print "`Impossible`" instead.

Sample Input

```
4 2 6
95050 0 1
0 1 2
1 104729 1
95050 2 1
2 3 3
3 104729 1
4 1 6
95050 0 1
0 1 2
1 104729 1
95050 2 1
2 3 3
3 104729 1
4 2 7
95050 0 1
0 1 2
1 104729 1
95050 2 1
2 3 3
3 104729 1
2 1 5
0
```

Sample Output

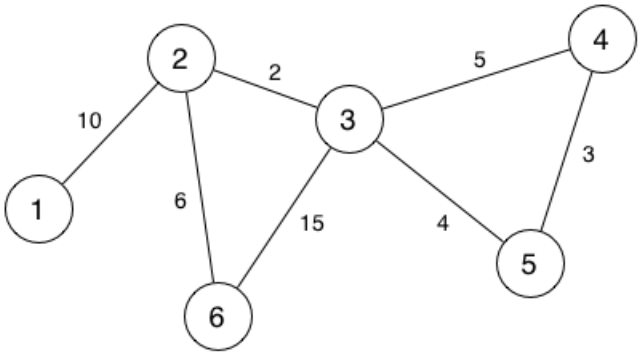
```
2.5
Impossible
3.0
```

I: Cost

You are given an undirected graph with N verticies and M edges, where the weights are unique.

There is a function `Cost(u,v)`, which is defined as follows:

While there is a path between vertex u and v, delete the edge with the smallest weight. `Cost(u,v)` is the sum of the weights of the edges that were deleted in this process.



For example, from the graph above (same as the sample input), `Cost(2,6)` is $2 + 3 + 4 + 5 + 6 = 20$.

Given an undirected graph, your task is to calculate the sum of `Cost(u,v)` for all vertices u and v, where $u < v$. Since the answer can get large, output the answer modulo 10^9.

Input

There are many test cases. The first line of each test case consists of two integers, N and M. ($1 \leq N \leq 100,000$, $0 \leq M \leq 100,000$). The next M lines consists of three integers, u, v and w. This means that there is an edge between vertex u and v with weight w. ($1 \leq u, v \leq N$, $1 \leq w \leq 100,000$).

Output

Output the sum specified in the problem statement.

Sample Input

```
6 7
1 2 10
2 3 2
4 3 5
6 3 15
3 5 4
4 5 3
2 6 6

2 1
1 2 1
```

Sample Output

```
256
1
```

J: Silly Sort

Your younger brother has an assignment and needs some help. His teacher gave him a sequence of numbers to be sorted in ascending order. During the sorting process, the places of two numbers can be interchanged. Each interchange has a cost, which is the sum of the two numbers involved.

You must write a program that determines the minimal cost to sort the sequence of numbers.

Input

The input file contains several test cases. Each test case consists of two lines. The first line contains a single integer n ($n > 1$), representing the number of items to be sorted. The second line contains n different integers (each positive and less than 1000), which are the numbers to be sorted.

The input is terminated by a zero on a line by itself.

Output

For each test case, the output is a single line containing the test case number and the minimal cost of sorting the numbers in the test case.

Place a blank line after the output of each test case.

Sample Input

```
3
3 2 1
4
8 1 2 4
5
1 8 9 7 6
6
8 4 5 3 2 7
0
```

Sample Output

```
Case 1: 4

Case 2: 17

Case 3: 41

Case 4: 34
```