Problem N. Nale Sort

Input file:	stdin
Output file:	stdout
Time limit:	3 seconds
Memory limit:	512 megabytes

Nale developed a sorting algorithm described by the following pseudocode:

```
nale_sort(array nums) {
    if (nums.size <= 1) {
        return nums;
    }
    pivot = nums[0];
    array less;
    array more;
    for (i = 1; i < nums.size; ++i) {
        // Comparison
        if (nums[i] < pivot) {
            less.append(nums[i]);
        }
        else {
            more.append(nums[i]);
        }
    }
    sorted_less = nale_sort(less);
    sorted_more = nale_sort(more);
    ans = sorted_less + pivot + sorted_more;

    return ans;
}
```

We consider a comparison to be any time some `nums[i]` is compared with `pivot`.

You must solve q queries where each query consists of some permutation of $1, 2, \ldots, n$. Please find out how many comparisons are needed to sort the array with `nale_sort`.

Input

The first line contains a single integer denoting q (the number of queries).

Each queries has two lines. The first line contains an integer n, and the second line contains $p_1, p_2, \ldots, p_n$ where $1 \leq p_i \leq n$ and all p_i's are distinct.

Constraints

- $1 \leq q \leq 10^5$
- $1 \leq n \leq 10^5$
- $\sum n \leq 10^6$

Output

For each query, print the number of comparisons needed in one line.

Example

stdin	stdout
1 5 4 2 1 3 5	6
3 1 1 4 4 3 2 1 3 2 1 3	0 6 2

Explanation

For example 1:

We perform the following 6 queries:

For $[4, 2, 1, 3, 5]$, the sequence of sorting operations looks like this:

1. Run `nale_sort` on $[4, 2, 1, 3, 5]$: Compare $pivot = 4$ with $2, 1, 3$, and 5 for a total of 4 comparisons. We're then left with $less = [2, 1, 3]$ and $more = [5]$; we only need to continue sorting less, as more is sorted with respect to itself because it only contains one element.
2. Run `nale_sort` on $less = [2, 1, 3]$: Compare $pivot = 2$ with 1 and 3 for a total of 2 comparisons. We're then left with $less = [1]$ and $more = [3]$, so we stop sorting.

Thus we sorted $[4, 2, 1, 3, 5]$ in $4 + 2 = 6$ comparisons.

Problem O. Oxx Playing Pokemon Go

Input file:	stdin
Output file:	stdout
Time limit:	2 seconds
Memory limit:	512 megabytes

Oxx is still a passionate player of the world-renowned game Pokemon Go. Recently, he decided to organize a competition in catching Pokemon. It will be held in Ilica street in Zagreb, and the main sponsor is his friend Xxo. The reward is, of course, candy!

Ilica is, as we all know, the longest street in Zagreb. There are N houses on the same side of the street, and each house has a house number between 1 and N. The competition begins at house number K.

Before the competition, Oxx looked at the map and saw M Pokemon. Each Pokemon is located at its (distinct) house number A_i, is valued at B_i candy, and can be caught only in the next $T_i - 1$ seconds, after which it disappears from the map at the time T_i and is impossible to catch since then.

Oxx can visit one house number per second. Also, when he catches a Pokemon, that Pokemon disappears from the map.

Since Oxx really likes candy, he asked for your help.

Help him and determine the maximal amount of candy he can get!

Input

The first line of input contains integers N, K ($1 \le K \le N \le 1000$) and M ($1 \le M \le 100$), the number of houses, the starting house number and the number of Pokemon.

Each of the following M lines contains integers A_i ($1 \le A_i \le N$), B_i ($1 \le B_i \le 100$) and T_i ($1 \le T_i \le 2\,000$) from the task.

In the input data, the Pokemon will always be in a strictly ascending order by house number A_i.

Output

You must output the required maximal amount of candy from the task.

Example

stdin	stdout
10 5 4 1 30 4 3 5 7 7 10 12 9 100 23	115
20 8 7 1 35 14 4 57 1 6 32 2 9 94 28 14 78 8 15 8 1 17 55 3	172

Problem P. Prefix Free Code

Input file:	stdin
Output file:	stdout
Time limit:	2 seconds
Memory limit:	512 megabytes

Consider n initial strings of lowercase letters, where no initial string is a prefix of any other initial string. Now, consider choosing k of the strings (no string more than once), and concatenating them together. You can make this many such composite strings:

$$n \times (n-1) \times (n-2) \times ... \times (n-k+1)$$

Consider sorting all of the composite strings you can get via this process in alphabetical order. You are given a test composite string, which is guaranteed to belong on this list. Find the position of this test composite string in the alphabetized list of all composite strings, modulo $10^9 + 7$. The first composite string in the list is at position 1.

Input

Each input will consist of a single test case. Note that your program may be run multiple times on different inputs.

Each test case will begin with a line with two integers, first n and then k ($1 \leq k \leq n$), where n is the number of initial strings, and k is the number of initial strings you choose to form composite strings. The upper bounds of n and k are limited by the constraints on the strings, in the following paragraphs. Each of the next n lines will contain a string, which will consist of one or more lower case letters $a..z$. These are the n initial strings. It is guaranteed that none of the initial strings will be a prefix of any other of the initial strings.

Finally, the last line will contain another string, consisting of only lower case letters $a..z$. This is the test composite string, the position of which in the sorted list you must find. This test composite string is guaranteed to be a concatenation of k unique initial strings.

The sum of the lengths of all input strings, including the test string, will not exceed 10^6 letters.

Output

Output a single integer, which is the position in the list of sorted composite strings where the test composite string occurs. Output this number modulo $10^9 + 7$.

Example

stdin	stdout
5 3 a b c d e cad	26
2 2 lubenwei niubi lubenweiniubi	1

Problem Q. Query On Tree

Input file:	stdin
Output file:	stdout
Time limit:	2 seconds
Memory limit:	512 megabytes

You are given a tree with N nodes and each node's value val_i is initially .

Let's denote the path from node u to node v like this: $p_1, p_2, p_3, \ldots, p_k$, where $p_1 = u$ and $p_k = v$, and p_i and p_{i+1} are connected.

The problem asks you to operate the following two types of queries on the tree:

- 1 u v x Add x to val_{p_1}, $2x$ to val_{p_2}, $3x$ to val_{p_3}, ..., kx to val_{p_k}.
- 2 u v print the sum of the nodes' values on the path between u and v at modulo $10^9 + 7$.

Input

First line cosists of two integers N and Q seperated by a space.

The following $N - 1$ lines each contains two integers which denote the undirectional edges of the tree.

The following Q lines contains one of the query types described above.

Note: Nodes are numbered by using 0-based indexing.

Constraints

- $1 \leq N, Q \leq 50000$
- $0 \leq x < 10^9 + 7$

Output

For every query of second type print a single integer.

Example

stdin	stdout
3 2 0 1 1 2 1 0 2 1 2 1 2	5

Explanation

After the first type of query, $val_0 = 1, val_1 = 2, val_2 = 3$. Hence the answer of the second query is $(2 + 3) \bmod (10^9 + 7) = 5$.

Problem R. Rabbit's Howse

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	512 megabytes

Pooh, his face covered with honey and all sticky, thanks Rabbit and eats leftover honey on his stomach, which is now extremely round and full. He tries to leave through Rabbit's front door, but has become extremely large from the vast amount of honey he has eaten — so fat that Pooh gets stuck in Rabbit's front door. Rabbit tries to free Pooh by pushing his over-sized bottom but it isn't any use. He goes round to the front of the house to face Pooh's head, and tells Pooh, truthfully, that he has eaten too much, and as a result, he has grown too fat for Rabbit's front door.

Input

Christopher Robin, Rabbit, and Eeyore arrive and try to help Pooh by pulling him out with all their strengths. You are given the strength of Christopher Robin, Rabbit and Eeyore, c, r and e respectively, you are asked whether the sum of their strength, i.e., $c + r + e$, is enough to pull Pooh out. It's also estimated that the minimum strength required to pull him out is m.

The only line of the input contains c, r, e, m ($1 \leq c, r, e, m \leq 100$), separated with space.

Output

So in case you don't have the time to read the story, if $c + r + e \geq m$, you should output "`Pooh out`", otherwise, output "`Stuck in the howse forever`".

Example

stdin	stdout
1 2 3 6	Pooh out
1 2 3 7	Stuck in the howse forever

Problem S. Substring Diff

Input file:	stdin
Output file:	stdout
Time limit:	2 seconds
Memory limit:	512 megabytes

In this problem, we'll use the term "longest common substring" loosely. It refers to substrings differing at some number or fewer characters when compared index by index. For example, 'abc' and 'adc' differ in one position, 'aab' and 'aba' differ in two.

Given two strings and an integer k, determine the length of the longest common substrings of the two strings that differ in no more than k positions.

For example, $k = 1$. Strings $s_1 = abcd$ and $s_2 = bbca$. Check to see if the whole string (the longest substrings) matches. Given that neither the first nor last characters match and $2 > k$, we need to try shorter substrings. The next longest substrings are $s'_1 = [abc, bcd]$ and $s'_2 = [bbc, bca]$. Two pairs of these substrings only differ in 1 position: $[abc, bbc]$ and $[bcd, bca]$. They are of length 3.

Input

The first line of input contains a single integer, T, the number of test cases follow. Each of the next T lines contains three space-separated values: an integer k and two strings, s_1 and s_2.

Constraints

- $1 \leq T \leq 10$
- $0 \leq k \leq |s_1| = |s_2|$
- $1 \leq |s_1| = |s_2| \leq 1500$

Output

For each test case, output a single integer which is the length of the maximum length common substrings differing at k or fewer positions.

Example

stdin	stdout
3 2 tabriz torino 0 abacba abcaba 3 helloworld yellomarin	4 3 8

Explanation

First test case: If we take "briz" from the first string, and "orin" from the second string, then the number of mismatches between these two substrings is equal to 2 and their lengths are 4.

Second test case: Since $k = 0$, we should find the longest common substring, standard definition, for the given input strings. We choose "aba" as the result.

Third test case: We can choose "hellowor" from first string and "yellomar" from the second string.

Problem T. Transform IP Database

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	512 megabytes

Having a heavy load on its servers, Infiniteee keeps a daily log of the traffic to its server for performance optimization and statistical purposes. The IP addresses in the log for each day is stored in a database either in the IP-range format or CIDR format, both explained below.

The IP database software uses a new version of the Internet Protocol (IP) called IPv5. IPv5 provides more IP addresses by extending the number of address bits from 32 in the IPv4 to 40. Precisely, each IP address in the IPv5 is a 40-bits numerical label assigned to each device connected to the Internet. Each IP address can be represented as a sequence of 5 numbers, called bytes, each having a decimal value ranging from 0 to 255. IP addresses are often written in the dot-decimal notation, for example "134.172.16.254.1". The notation consists of five bytes of the IP address expressed in decimal and separated by periods. Note that in this notation the leading zero bytes are not removed. For example, the correct representation of an address with numerical value 0 is "0.0.0.0.0".

Classless Inter-Domain Routing (CIDR) is a way of specifying a range of IP addresses. A CIDR looks like a normal IP address except that it ends with a slash followed by a number. The CIDR y/x in which y is an IP address and x is an integer from 0 to 40 (inclusive), shows a range of addresses whose x leftmost bits are equal to the x leftmost bits of y. The remaining bits are free to be either 0 or 1. To guarantee a unique representation for each range, the $40 - x$ rightmost bits of y should be equal to 0.

The IP-range format is another way of specifying a range of IP addresses. A range in this format is represented by its first and last address in dot decimal notation, separated by a dash ($-$). The first address is not larger than the last address assuming IP addresses are 5-digit numbers in the base 256. For example, "128.192.168.200.0/32" is a CIDR in which the first 32 bits (4 bytes, "128.192.168.200") are the same for all addresses in the range, and only the last byte can be different. The same range can be represented in the IP-range format by "128.192.168.200.0 $-$ 128.192.168.200.255".

Due to attracting many customers from all around the world and continuous service for many years, the IP database is getting larger and larger. Thus, Infiniteee plans to reduce the number of entries in the database by representing the existing ranges using the minimum number of CIDRs. The new CIDRs should not include any IP address that does not belong to the IP database. Your task is to solve this challenging problem.

Input

There are multiple test cases in the input. For each test case, the first line contains an integer n, the number of IP ranges ($1 \le n \le 100$) in the IP database. Each of the next n lines contains an IP range, either in the CIDR format or in the IP-range format. The input terminates with a line containing 0 which should not be processed.

Output

For each test case, print the minimum number of CIDRs which represent the whole IP database; followed by the list of CIDRs in an increasing order of IP values.

Example

stdin	stdout
2 128.192.168.10.0/32 128.192.168.11.0-128.192.168.11.255 1 128.192.168.200.0-128.192.168.200.10 0	1 128.192.168.10.0/31 3 128.192.168.200.0/37 128.192.168.200.8/39 128.192.168.200.10/40

Problem U. Ultmaster Dividing The Cake

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	512 megabytes

Ultmaster and Zerol are party-friends. Yesterday they threw a big party and today they have a partially eaten cake that they want to divide. Since people were careless when cutting themselves a slice, the cake is now shaped as a prism with its top and bottom faces being the same simple convex polygon.

To add some fun to the process of dividing the cake, they came up with the following game. Ultmaster chooses a vertex v of the top face of the cake. Zerol chooses another vertex w of the top face that is not adjacent to v. Then, they cut the cake into two pieces by extending downwards the segment $v - w$, so as to obtain two separate pieces of cake, each in the shape of a prism. Finally, Ultmaster chooses the piece that she prefers, and Zerol gets the other one. Zerol immediately saw that this system gives Ultmaster an advantage. Zerol wants to know exactly how much of an advantage Ultmaster has.

You are given a polygon that represents both the top and bottom faces of the cake. The height of the cake is 2, so the volume of a piece of cake is 2 times the area of its top face. Assuming the cake is divided as explained, and that both of them make their decisions to maximize the volume of the piece they have at the end, compute the volume of the piece each girl will get.

Input

The first line contains an integer n representing the number of vertices of the polygonal top face of the cake ($4 \leq n \leq 10^5$). Each of the next n lines describes a vertex of the polygon with two integers x and y, indicating the coordinates of the vertex in the xy plane ($-10^8 \leq x, y \leq 10^8$). Vertices are given in counter clockwise order and define a simple convex polygon. No three points in the input are collinear.

Output

Output a line with two integers representing the volume of the piece Carol and Carla will get, in that order, if both make their decisions optimally.

Example

stdin	stdout
5 0 0 3 0 3 1 2 2 0 1	7 2
6 0 1 1 0 2 0 3 1 2 2 0 2	6 3
4 -100000000 -100000000 100000000 -100000000 100000000 100000000 -100000000 100000000	40000000000000000 40000000000000000
4 -99999995 -100000000 100000000 -100000000 100000000 99999995 -100000000 100000000	39999999999999975 39999998000000025

Problem V. Viva La Vida

Input file:	stdin
Output file:	stdout
Time limit:	2 seconds
Memory limit:	512 megabytes

Steve has been the king of a huge kingdom. This kingdom consists of N people, labeled from 1 to N, and Steve is labeled 1. The kingdom is structured so that each citizen (except Steve) has a master, and we say that citizen is an slave to that master.

Each master can have multiple slaves, but still pledges loyalty to their master. This holds for everyone except Steve, who is at the top of the pyramid and doesn't have a master, but has his slaves.

When Steve gets a new hat and wants a sacrifice, he then delegates that order to his slave with the minimal number. This slave then delegates the order to their slave with the minimal number, and this process repeats until the order is given to an unlucky person without an slave, shouting "Viva la vida!".

This is when the real problem begins. The person that dies for the hat gets paid 1 coin, the master of that person gets 2 coins, the master of that person gets 3 coins, and so on, all the way to Steve, who gets as many coins as there are people in this sequence. After that, the dead citizen goes to heaven and never comes back again.

When it comes to doing the next order in the kingdom, there is a person less, so maybe the paychecks are less, but the work must continue. Orders are piling up, so the whole procedure (assigning a new order, doing it, dividing paychecks and the person doing the order quitting) repeats until Steve is left alone in the kingdom and dies for his own new hat.

Of course, Steve will have amassed quite a fortune until then, but he also wants to know how much money each of the citizens earned, though they all die afterall.

Input

The first line of input contains the positive integer N ($2 \le N \le 2 \cdot 10^5$), the number of citizens (including Steve).

The following line contains $N-1$ positive integers a_i ($1 \le a_i \le i$, $2 \le i \le N$), where a_i denotes the master of citizen i.

Output

You must output a single line consisting of N numbers, the i_{th} number corresponding to the amount of money earned by the i_{th} citizen.

Example

stdin	stdout
3 1 1	5 1 1
5 1 2 2 4	13 8 1 3 1

Problem W. Weight Of Subsequence

Input file:	stdin
Output file:	stdout
Time limit:	2 seconds
Memory limit:	512 megabytes

A subsequence of a sequence is a sequence which is obtained by deleting zero or more elements from the sequence.

You are given a sequence A in which every element is a pair of integers, i.e., $A = [(a_1, w_1), (a_2, w_2), ..., (a_N, w_N)]$.

For a subseqence $B = [(b_1, v_1), (b_2, v_2), ..., (b_M, v_M)]$ of the given sequence:

We call it increasing if for every i $(1 \leq i < M)$, $b_i < b_{i+1}$. Then we define $weight(B) = v_1 + v_2 + \cdots + v_M$.

Given a sequence, output the maximum weight formed by an increasing subsequence.

Input

The first line of input contains a single integer T. T test-cases follow. The first line of each test-case contains an integer N. The next line contains $a_1, a_2, \ldots, a_N$ separated by a single space. The next line contains $w_1, w_2, \ldots, w_N$ separated by a single space.

Output

For each test-case output a single integer: The maximum weight of increasing subsequences of the given sequence.

Constraints

- $1 \leq T \leq 5$
- $1 \leq N \leq 150000$
- $1 \leq a_i \leq 10^9$
- $1 \leq w_i \leq 10^9$

Example

stdin	stdout
2 4 1 2 3 4 10 20 30 40 8 1 2 3 4 1 2 3 4 10 20 30 40 15 15 15 50	100 110

Explanation

In the first sequence, the maximum size increasing subsequence is 4, and there's only one of them. We choose $B = [(1, 10), (2, 20), (3, 30), (4, 40)]$, and we have $weight(B) = 100$.

In the second sequence, the maximum size increasing subsequence is still 4, but there are now 5 possible subsequences:

- $[(1, 10), (2, 20), (3, 30), (4, 40)]$

- $[(1, 10), (2, 20), (3, 30), (4, 50)]$
- $[(1, 10), (2, 20), (3, 15), (4, 50)]$
- $[(1, 10), (2, 15), (3, 15), (4, 50)]$
- $[(1, 15), (2, 15), (3, 15), (4, 50)]$

Of those, the one with the greatest weight is $B = [(1, 10), (2, 20), (3, 30), (4, 50)]$, with $weight(B) = 110$.

Please note that this is not the maximum weight generated from picking the highest value element of each index. That value, 115, comes from $[(1, 15), (2, 20), (3, 30), (4, 50)]$, which is not a valid subsequence because it cannot be created by only deleting elements in the original sequence.

Problem X. X Window

Input file:	stdin
Output file:	stdout
Time limit:	2 seconds
Memory limit:	512 megabytes

You wake up and find yourself being X Server. The painting operations are waiting! Hurry up!

You are given N points in the coordinate system. They need to be covered with one or more windows (rectangles), such that the following conditions are met:

1. The sides of each window are parallel with the coordinate axes.
2. The center of each window is in the origin, i.e. point $(0, 0)$.
3. Each given point is located either inside of the window or on its boundaries.

Of course, it is possible to cover all the points using only one window, but this window could have a very large surface area. Our goal is to find a selection of required windows such that the sum of their surface areas is minimal to minimize your work.

Input

The first line of input contains the integer N ($1 \le N \le 5000$), the number of points.

Each of the following N lines contains two integers X and Y ($-50\,000\,000 \le X, Y \le 50\,000\,000$, $XY \neq 0$), the coordinates of each point.

Output

You must output the required minimal sum of surface areas of the windows.

Example

stdin	stdout
2 1 1 -1 -1	4
3 -7 19 9 -30 25 10	2080
6 1 20 3 17 5 15 8 12 9 11 10 10	760

Problem Y. You Only Live Once

Input file:	stdin
Output file:	stdout
Time limit:	2 seconds
Memory limit:	512 megabytes

The Byte Land is on fire due to the attack of the virtual enemy. Several places are already on fire and the fire is spreading fast to other places. You are the only person remaining alive in the war with the virtual enemy, trying to rescue yourself by reaching to the only helicopter in the Byte Land.

Suppose the Byte Land is represented by an $n \times m$ grid. At the initial time, some grid cells are on fire. If a cell catches fire at time x, all its 8 vertex-neighboring cells will catch fire at time $x + k$. If a cell catches fire, it will be on fire forever. At the initial time, you stand at cell s and the helicopter is located at cell t. At any time x, you can move from its current cell to one of four edge-neighboring cells, located at the left, right, top, or bottom of its current cell if that cell is not on fire at time $x + 1$. Note that each move takes one second. Your task is to write a program to find the shortest path from s to t avoiding fire.

Hurry up! You can only live once!

Input

There are multiple test cases in the input. The first line of each test case contains three positive integers n, m and k ($1 \le n, m, k \le 100$), where n and m indicate the size of the test case grid $n \times m$, and k denotes the growth rate of fire. The next n lines, each contains a string of length m, where the j-th character of the i-th line represents the cell (i, j) of the grid. Cells which are on fire at time 0, are presented by character "`f`". There may exist no "`f`" in the test case. The helicopter and yourself are located at cells presented by "`t`" and "`s`", respectively. Other cells are filled by "`-`" characters. The input terminates with a line containing "`0 0 0`" which should not be processed.

Output

For each test case, output a line containing the shortest time to reach t from s avoiding fire. If it is impossible to reach t from s, write "Impossible" in the output.

Example

stdin	stdout
7 7 2 f------ -f---f- ----f-- ------- ------f ---s--- t----f- 3 4 1 t--f --s- ---- 2 2 1 st f- 2 2 2 st f- 0 0 0	4 Impossible Impossible 1

Problem Z. Zerol And KBlack

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 second
Memory limit:	256 megabytes

Zerol and KBlack are bored on a rainy day and decide to pass the time by creating a new game having the following rules:

The game starts with two n-sized integer arrays, A and B, and is played by two players, P_1 and P_2.

The players move in alternating turns, with P_1 always moving first. During each move, the current player must choose an integer, i, such that $0 \leq i \leq n-1$. If the current player is P_1, then receives A_i points; if the current player is P_2, then receives B_i points.

Each value of i can be chosen only once. That is, if a value of i is already chosen by some player, none of the player can re-use it. So, game always ends after n moves.

The player with the maximum number of points wins.

The arrays A and B are accessible to both the players P_1 and P_2. So the players make a optimal move at every turn.

Given the values of n, A, and B, can you determine the outcome of the game? Print `First` if P_1 will win, `Second` if P_2 will win, or `Tie` if they will tie. Assume both players always move optimally.

Input

The first line of input contains a single integer, T, denoting the number of test cases. Each of the $3T$ subsequent lines describes a test case. A single test case is defined over the following three lines:

1. An integer, n, denoting the number of elements in arrays A and B.
2. n space-separated integers, $A_0, A_1, \ldots, A_{n-1}$, where each describes the element at index i of array A.
3. n space-separated integers, $B_0, B_1, \ldots, B_{n-1}$, where each describes the element at index i of array B.

Constraints

- $1 \leq T \leq 10$
- $1 \leq n \leq 1000$
- $1 \leq A_i, B_i \leq 10^5$

Output

For each test case, print one of the following predicted outcomes of the game on a new line:

Print `First` if P_1 will win. Print `Second` if P_2 will win. Print `Tie` if the two players will tie.

Example

stdin	stdout
3 3 1 3 4 5 3 1 2 1 1 1 1 2 2 2 3 3	First Tie Second

Explanation

Test Case 0: $A = \{1, 3, 4\}$, $B = \{5, 3, 1\}$ The players make the following moves:

1. P_1 chooses $i = 2$ and receives 4 points.
2. P_2 chooses $i = 0$ and receives 5 points. Note that P_2 will not choose $i = 1$, because this would cause P_1 to win.
3. P_1 chooses $i = 1$ (which is the only remaining move) and receives 3 points.

As all $n = 3$ moves have been made, the game ends. P_1's score is 7 points and P_2's score is 5 points, so P_1 is the winner and we print `First` on a new line.

Test Case 1: $A = \{1, 1\}$, $B = \{1, 1\}$ Because both players will only make 1 move and all possible point values are 1, the players will end the game with equal scores. Thus, we print `Tie` on a new line.

Test Case 2: $A = \{2, 2\}$, $B = \{3, 3\}$ Because both players will only make 1 move and all the possible point values for P_2 are greater than all the possible point values for P_1, P_2 will win the game. Thus, we print `second` on a new line.