



Drawing Rooted Binary Trees	3
Organizational Chart	6
Barter System.....	8
Troll Coder	9
Troll Coder - Escape	10
PIEEXMan	12
Xtreme Fake Coins.....	17
Magic Spell.....	19
Bear Sums	20
Infinite snakes and ladders	22
Lemons	26
Make Distinct	28
Telescope Scheduling	29
Xplore	31
The Gold-diggers	33
Tweedledee's Split Brackets	37
Tweedledum's Nested Brackets.....	38
Friendly Sequences.....	40
Gotta ~ Catch 'Em All.....	41
Product Rectangle	43
Tree Fun	44
Factorial Zeros.....	46

Bit Soccer	47
Barrett Reduction	49
Feedback Challenge.....	50

Drawing Rooted Binary Trees

Time limit: 1280 ms

Memory limit: 264 MB

Binary trees are a common data structure and understanding some basic operations is important. For this problem, you will play with traversals and tree drawing strategies.

An ASCII Representation of a Tree

When you are debugging something that uses a **binary tree** is often valuable to be able to get a quick description of the tree to the console (in characters) so you can check its structure. The basic strategy is to use spaces to position the nodes so that you can infer the tree's connectivity. For example, a tree that is just a root with two children can be "drawn" as:

1	a
2	b c
3	

These types of drawings follow these rules:

- all nodes in a left subtree must be to the left of the root
- all nodes in a right subtree must be to the right of the root
- the root must be between the two subtrees
- no other node will be in the same column as the root
- every node must be as far left as possible without breaking the other rules

Standard input

For this problem, all node labels are single characters and every tree will contain at least one node. Each instance of the problem requires two lines of input:

- the first line will be the infix traversal of the tree
- the second line will be the prefix traversal of the tree

One run of the problem may contain many instances of the problem.

Standard output

For each instance of the problem, you must output the ASCII representation of the tree (trailing characters in a line will be ignored). Multiple instance of the problem should follow immediately (no blank lines between them).

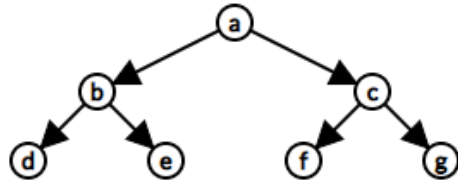
Constraints and notes

- All nodes are represented by lowercase English letters

Input	Output	Explanation
<pre> bac abc </pre>	<pre> a b c </pre>	<pre> graph TD a((a)) --> b((b)) a --> c((c)) </pre>
<pre> ba ab </pre>	<pre> a b </pre>	<pre> graph TD a((a)) --> b((b)) </pre>
<pre> acb abc </pre>	<pre> a b c </pre>	<pre> graph TD a((a)) --> b((b)) b --> c((c)) b --> </pre>
<pre> acbd abcd cba abc </pre>	<pre> a b c d a b c </pre>	<pre> graph TD a1((a)) --> b1((b)) b1 --> c1((c)) b1 --> d1((d)) a2((a)) --> b2((b)) b2 --> c2((c)) </pre>

dbeafcg
abdecfg

a
b c
d e f g



Organizational Chart

Time limit: 4880 ms

Memory limit: 264 MB

The new Chief Executive Officer (CEO) at Big Tech wants to gather information on all of the divisions in the company. The CEO gathered all top management and asked them to report on the number of employees in each of the top level divisions. These managers in turn asked all of their subordinates for the number of employees in the child divisions of these top level divisions, and so on.

You have been given the organizational chart and the reported numbers, as well as a list of queries for the sizes of different divisions. However, some of these numbers are missing. Your task is to provide lower and upper bounds for each of the queries.

Standard input

The input begins with two integers N and Q .

Next come N lines each describing all divisions in the following format:

```
1 [division_name] [parent_division] [size1] [size2]
2 where
3   [division_name] is a child of [parent_division]
4   [size1] is the number of employees in [division_name] but not any child
   division below it in the corporate hierarchy.
5   [size2] is the number of employees in [division_name] including all
   divisions below it in the corporate hierarchy.
6 If there is no parent, the [parent_division] will be equal to "NONE".
7 If a size is missing, the size value will be set to 0.
8
```

Next come Q lines each describing each query in the form:

```
1 [division_name] [type]
2 where
3   [division_name] is a division name that exists in the corporate hierarchy,
   and
4   [type] is either 1 or 2
5
```

Output

For each query, your program should output a single line with two integers separated by a space.

For type 1 queries, you are to output lower and upper bounds on the number of employees in that division, **not including** divisions below it in the corporate hierarchy.

For type 2 queries, you are to output lower and upper bounds on the number of employees in that division **including** divisions below it in the corporate hierarchy.

Constraints and notes

- $1 \leq N, Q \leq 2 * 10^5$
- The maximum number of employees in the organization is $6 * 10^9$. (There is a reason the company is called Big Tech).
- Each division name is unique and is made up only of lowercase letters, numbers, and the underscore.
- The corporate hierarchy is a **rooted tree**.
- Every division must have at least one employee who is not an employee of a child division.
- All employees of a child division are also employees of the parent division, but not vice versa.

Input	Output	Explanation
<pre> 10 3 company NONE 10 100 engineering company 5 30 team1 engineering 10 10 team2 engineering 0 0 team2part1 team2 5 5 team2part2 team2 3 3 manufacturing company 40 40 finance company 0 0 accounting finance 0 0 sales finance 0 5 company 1 team2 2 accounting 1 </pre>	<pre> 10 10 15 15 1 14 </pre>	

The sample input corresponds to the organizational chart above.

The first query is for the size of **company**, not including its subdivisions, which is given as 10 in the input. Therefore, both the lower bound and upper bound are 10.

The second query is for the size of **team2** and its subdivisions. There are a total of 30 employees in the **engineering** and its portion of the hierarchy. Five are in the **engineering** and 10 are in **team1**. Therefore, **team2** and its subdivisions must have exactly 15 people.

For the last query, we know that there must be at least one person in the **accounting** division, not including its subdivisions. For the upper bound, we know that there are 20 people in the **finance** portion of the hierarchy. Of these, at least 6 must be in **finance and sales**, so the upper bound for **accounting** is 14.

Barter System

Time limit: 1280 ms

Memory limit: 264 MB

In a world where currency is not available, people fulfill their necessities by exchanging commodities among them. However, not all commodities are equal.

Exchange rate of commodities A to B is defined as the number of units of B one can get with one unit of A .

Given exchange rates of a set of pairs of commodities as input, answer a set of queries by finding the exchange rate of a specific pair.

Standard input

The first line of the input is an integer N which indicates the number of given exchange rates to follow. The next N lines consists of A, B, r triplets where A and B are the commodities and r is the exchange rate such that $A = r \cdot B \pmod{998244353}$.

The next line contains another integer Q which represents the number of queries. The following Q lines consists of a pair of commodities K and L .

Standard output

For each of the Q queries you need to find r such that $K = r \cdot L$.

It can be shown that r can be represented as $\frac{X}{Y}$, where X and Y are coprime integers and $X \neq 0 \pmod{998244353}$. For each query print $X \cdot Y^{-1} \pmod{998244353}$.

If the exchange rate can not be computed using the given information, print `-1`.

Constraints and notes

- $1 \leq N, Q \leq 2 \cdot 10^4$
- $1 \leq |A|, |B| \leq 50$
- $1 \leq r < 998244353$
- All exchange rates are consistent

Input	Output	Explanation
3 Rice Bean 399297742 Bean Beef 598946612 Banana Apple 698771048 3 Rice Beef Beef Banana Rice Rice	279508419 -1 1	Rice = $399297742 \cdot 598946612$ Beef, which is 279508419. There is not enough information to convert from Beef to Banana, so <code>-1</code> is printed.

Troll Coder

Time limit: 1280 ms

Memory limit: 264 MB

Interactive

You have found a huge treasure on an island, but the only access point to that island is a bridge guarded by a giant Troll! In order to cross the bridge, you have to guess a sequence of N bits by submitting queries. For each query, the Troll will tell you how many bits you guesses correctly until you guess the correct sequence.

Interaction

Your program must exchange information with the Troll by submitting queries and reading answers.

Note that you must flush the buffer so that the output reaches the Troll. [Here](#) we illustrate it for several languages.

At the beginning of each test case, the Troll will give you a single integer N which will represent the length of the sequence.

To submit a query, your program should output the letter Q followed by a space and by a binary sequence of length N with each bit separated by a space. After each query you will receive an integer denoting the number of correct bits. The last submission will be your final answer and it should start with an A followed by a space and by a binary sequence of length N with each bit separated by a space.

Constraints and notes

- This task is **NOT** adaptive
- $1 \leq N \leq 100$
- Your program can submit at most $N + 1$ queries before arriving at the correct answer.

Interaction	Explanation
6	This sequence has 6 bits and the solution is 101101.
	The first query is Q 0 0 0 0 0 0 0 which has 2 correct bits.
2	The correct answer is given in the form of A 1 0 1 1 0 1.
	A 1 0 1 1 0 1

Troll Coder - Escape

Time limit: 12080 ms

Memory limit: 264 MB

Interactive

The treasure is in your hands but there is only one way out of the island: the same way you used to get in! You have to cross the bridge again and the Troll is there waiting for you.

You are now faced with the same sequence guessing game, each sequence with exactly $N=100$. You have to solve every test with a maximum of $N+1$ queries to successfully cross the bridge, but the more queries you use the more treasure you will lose to the Troll.

Your final score for this challenge will be based on the average number of queries needed per test case. If you use $N+1$ queries on all the tests you will not score any points.

Interaction

At the beginning, you must read T , the number of test cases.

At the beginning of each test case, the Troll will give you a single integer N which will represent the length of the sequence.

To submit a query, your program should output the letter `Q` followed by a space and by a binary sequence of length N with each bit separated by a space. After each query you will receive an integer denoting the number of correct bits. The last submission will be your final answer and it should start with an `A` followed by a space and by a binary sequence of length N with each bit separated by a space.

Scoring

Assuming you provided a correct solution for all test cases, the scoring is computed as follows: let A be the average of number of `Q` queries over all cases, then your score will be $5 \cdot (80 - A)$. In case this is negative, you will score 0 points. In case this is greater than 100, you will score 100 points.

Constraints and notes

- This task is **NOT adaptive**
- You will be scored based on a single run of $T = 100$, with $N = 100$ for each test case.
- Your program can submit at most $N + 1$ queries before arriving at the correct answer.

Interaction	Explanation
2	For clarity, the example uses $N = 6$, but the actual test cases will consist solely of $N = 100$.
6	
Q 0 0 0 0 0 0	
2	A 1 0 1 1 0 1
6	A 0 0 1 0 1 1

PIEEXMan

Time limit: 1280 ms

Memory limit: 264 MB

Interactive

Everyone knows about the classic arcade game Pac-Man. This year, you're about to be playing in a special variation of it, called the PIEEXMan. In this variation PIEEXMan, our hero, munches his way around the maze collecting cherries, while Bob the Bear, our anti-hero, can't eat our hero, like the monsters in the original game, so instead he is looking to steal as many cherries as he can.

The maze is a 0-based $(2N + 1) \times (2M + 1)$ matrix of characters A . The cells are represented by the pairs (i, j) where i and j are odd integers. Moving throughout the cells can be done in the 4 directions: upward, downward, leftward and rightward; however it may be restricted by a wall. The character of the matrix directly in the direction you are facing encodes this information. For example, if $A_{i-1, j}$ contains a wall, then you cannot go from (i, j) to $(i - 2, j)$ (which is the cell located immediately upward).

You are playing in the role of PIEEXMan, while the judge is playing Bob. You start first.

Interaction

The judge will print two integers N and M , followed by a A , which is encoded as follows:

Character	Encoding	Where
1	PIEEXMan's initial position	only one, located in a cell
2	Bob the Bear's initial position	only one, located in cell
#	Wall	not in a cell
.	Empty	anywhere
@	Cherry	located in a cell

The moves you can make are as follows:

Character	Encoding
U	Move upward
D	Move downward
L	Move leftward
R	Move rightward
W	Wait

The judge makes the same type of moves. The game ends when the judge will print a move followed by the **X** character. In this case, you must end the interactions to get a proper verdict.

Download materials & maps

For this challenge there are 6 maps and numerous **judges** which whom you will compete to collect the cherries. There are 6 examples, each of them corresponding to one map and one **judge**. Not all **judges** are used in the examples.

You can download the [maps in txt and bmp format](#). The **txt** is the same as the one received in the interaction. The **image** is just a nice graphic representation of the map to help you better visualize it.

Scoring

Let A be the number of cherries you have collected, B be the number of cherries the judge has collected and C be the total number of cherries in the maze, then your score will be $5 \cdot (1 + \frac{A-B-1}{C+1})$.

Constraints and notes

- This task is **NOT adaptive**
- $1 \leq N, M \leq 30$
- A cherry may not be collected twice

Simulation

To see a simulation of a game follow the steps below

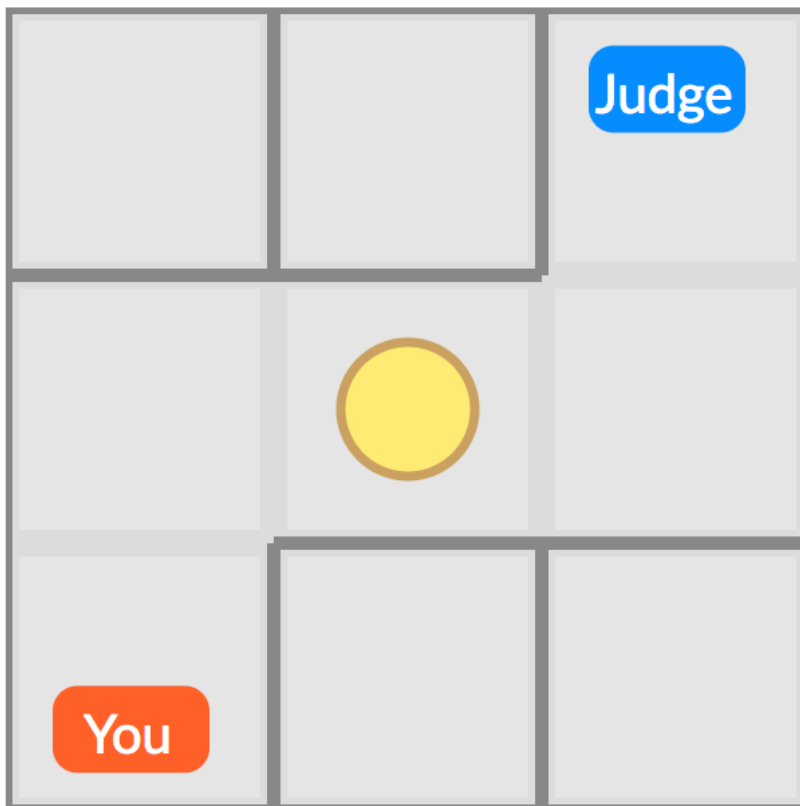
- Select a map that you would like to see a simulation for
- Open the **moves** link for the selected map
- Enter **all** the moves into the `moves` input box which is located below the simulation panel
- Open the **state** link for the selected map (the state is the same as the one that can be downloaded in the materials section)
- Enter the map description into the `state` input box which is located below the simulation panel
- Click the orange `Reload` button
- To see the simulation click the `play` button

Hints

- You can speed up the simulation replay (maximum of $16 \times$)
- You can enter your own simulation into the `moves` input box. Just beware that it **must** contain both your moves and the judge's moves.
- You can use the simulation to see a replay of your code on the examples. To do this, click the `Run Examples` button. Expand an example by clicking the `>` character in the `examples` panel. Carefully select the `interaction` and paste it into the `2` input boxes
- Beware that the `interaction` might look like the following (see below). You must split it accordingly before pressing the `Reload` button.

```
1 3 3
2 #####
3 #.#.#2#
4 #####.#
5 #.@.#
6 #.#####
7 #1#.#.#
8 #####UWRWX
```

Map Number	Example of Moves	Map state
0	Moves	State
1	Moves	State
2	Moves	State
3	Moves	State
4	Moves	State
5	Moves	State



You
0

Judge
0

Coins left: 1



Move: 0 of 4

Speed: 1x

Loaded

Moves:

1	U
2	W
3	R
4	W
5	X

State:

1	3 3
2	#####
3	#####2#
4	#####.#
5	#..@..#
6	#####
7	#1#####
8	#####

Xtreme Fake Coins

Time limit: 680 ms

Memory limit: 264 MB

Help IBM research's puzzlemaster to verify solutions for [May 2018's challenge](#).

There are NN coins, represented by the first NN capital letters of the English alphabet. Exactly two of them are counterfeit and have a slightly smaller weight. MM weightings using a double-pan balance scale have been performed, but they may not uniquely determine the pair of counterfeit coins.

Find all counterexamples of two pairs of coins $((a, b), (c, d))$ ($a < b, a \leq c, c < d, (a, b) \neq (c, d)$) whose weights are indistinguishable with respect to the M weightings.

Standard input

The first line contains two comma separated integers, N and M .

The next M lines contain two strings of disjoint subsets of the first N English capital letters, separated by a `-` sign.

There always is an equal amount of coins on both sides.

Standard output

List of lexicographically ordered counterexamples for the solution.

Each of them consists of two letters, an `=` sign and then another two letters.

A counterexample is a set of two pairs that cannot be distinguished by the set of M weightings.

Constraints and notes

- $0 \leq M \leq 10$
- $2 \leq N \leq 26$
- The counterexamples should be formed using only the first N letters

Input	Output	Explanation
11, 4 ABCDE–FGHIJ AHJ–FBD AGI–KCE BJ–IE	AC=AK AG=AH BC=BK BE=CD BE=DK BH=BJ BH=DJ BI=DG BJ=DJ CD=DK CE=EK CG=DH CH=EJ CI=EG CI=EI EG=EI FH=GK GH=IJ	<p>In the first weighting we are comparing ABCDE on the left pan and FGHIJ on the right; on the last weighting we compare BJ on the left with IE on the right. In the answer we give all the cases where the proposed solution does not work. For example the last line ($GH = IJ$) means that we can not distinguish between the case where G and H are counterfeited and the case where I and J are counterfeited. The reason is that in both cases the four results of the four weightings are the same:</p> <ul style="list-style-type: none"> • ABCDE > FGHIJ • AHJ < FBD • AGI < KCE • BJ = IE
15, 5 ABCDE–FGHIJ ACEGI–BDFHJ ABCKL–FDEM N EGOBH–IJLMN DEGKL–FMIBC	AB=BC AE=EK AF=CF AH=BG AI=CI AM=CM AM=CN BM=BN CM=CN DG=EH FH=FO FJ=FM FJ=FN FJ=JM FK=HJ FK=JO FM=FN FM=JM FN=JM HJ=JO IK=IL IM=IN JK=JL KM=LM KN=LN	<p>The first line in the solution ($AB = BC$) is because</p> <ul style="list-style-type: none"> • ABCDE < FGHIJ • ACEGI = BDFHJ • ABCKL < FDEM N • EGOBH < IJLMN • DEGKL > FMIBC
7, 3 ADE–BCG AG–BE AC–DG		<p>This solution is correct so there is no counterexample.</p>

Magic Spell

Time limit: 1280 ms

Memory limit: 264 MB

Write a program that gives the correct output for a legal input – to discover the correct function see the input/output examples.

Standard Input

The first line contains an integer N . Each of the following N lines will contain a legal string.

Standard output

Print the answer on the first line. It is guaranteed that it can be represented as a signed 64 integer.

Input	Output
<pre>2 yep momr momr rohjy drbrm</pre>	<pre>99999</pre>
<pre>3 yep pmr pmr momr drbrm yep xrtp gobr momr</pre>	<pre>111222111</pre>

Bear Sums

Time limit: 10880 ms

Memory limit: 264 MB

Mitsos the bear is challenging his brother Vangelis with a mathematical task. Mitsos provides a list of integers L and another integer value S , then he asks Vangelis to check if there are any two items in the list L whose sum is equal to the integer S .

Since Vangelis is a confident engineer, he decided to write a program that would do all the computations for him and skip the trouble of thinking. Your task is to help Vangelis write a program that reads the input list L and the integer S and produces either the solution to the problem or provides an error message when there is no solution available.

Standard input

On the first line there will be an integer number T (representing the number of test cases) followed by 2 input lines for each test case:

- On the first line of each test case, there will be 2 integers S and E , where S is the expected sum and E is the number of elements in the list.
- On the second line, there will be E integers separated by a space. Each integer represents an element of the list L . The elements are not sorted in any way and some could have the same value. In cases where the number E is 0, the second line will be empty.

All values for the elements of list L will be in the same range as the value S .

Standard output

For each test case you will have to write one line that contains:

- **If there is an unique solution:** Write two elements, x and y of the list L , separated by a single space, such that $x + y = S$ and $x \leq y$.
- **If there are multiple solutions:** Pick the first complete pair that appears on the list and provides the correct sum. Print the two list elements forming this pair in increasing order, as above.
- **If there is no solution:** Print the error message `!OK`.

Constraints and notes

- $1 \leq T \leq 1\,000$
- $-10^6 < S < 10^6$
- $0 \leq E \leq 2 \cdot 10^4$
- The sum of values of E is at most 10^7

Input	Output	Explanation
<pre> 6 8 4 1 2 4 4 8 4 1 2 7 9 8 4 1 2 8 9 8 4 4 5 3 4 8 4 4 1 1 8 8 4 -1 1 9 8 </pre>	<pre> 4 4 1 7 !OK 3 5 !OK -1 9 </pre>	<p>Line 1: Indicates the number of tests to follow, in this case we have 6 so we expect 12 more lines (two for each test).</p> <p>Line 2: Starts the first test. Two integers ($S = 8$ and $E = 4$). There will be 4 integers in the next line from which we want to find two of them that can produce the sum value of 8. In this case we have a solution if we sum the values 4 and 4 so this is what we print on the first line of the output.</p> <p>Line 4: Second test. Two integers ($S = 8$ and $E = 4$) again means that there will be 4 integers in the next line from which we want to find two of them that can produce the sum value of 8. In this case we have a solution as if we sum the values 1 and 7.</p> <p>Line 6: Third test. Two integers ($S = 8$ and $E = 4$). In this case we do not have a solution as there is no pair of values in the next line that can be added to obtain the wanted result and so we print <code>!OK</code> on the third line of the output.</p> <p>Line 8: Fourth test. In this case there are multiple solutions (5+3 and 4+4). The 5+3 option is chosen as this is the first complete pair in the sequence, <code>3 5</code> is printed to be in increasing order.</p>

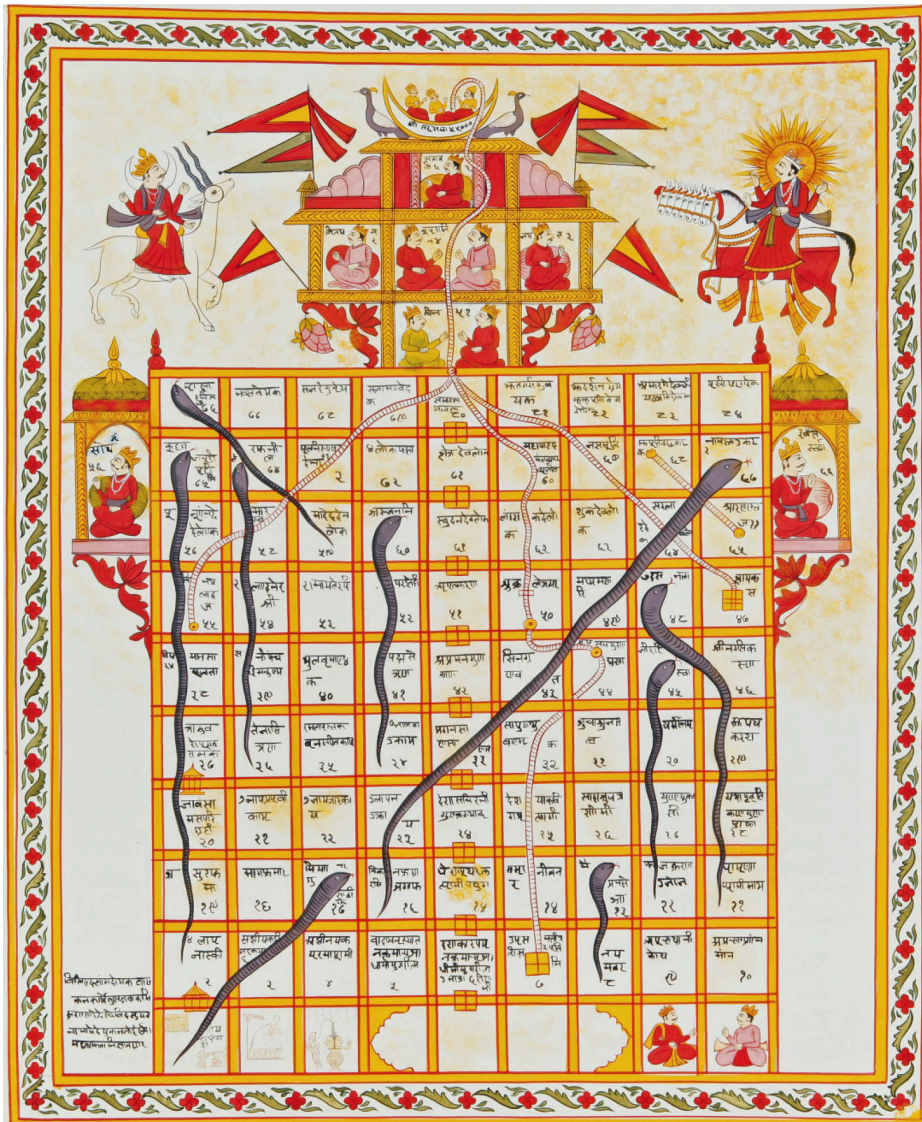
<pre> 10 12 4 5 0 8 9 13 4 10 0 6 1 17 3 2 9 4 0 8 2 7 6 5 3 9 8 0 4 0 4 0 7 7 8 10 9 7 4 6 2 13 3 6 7 1 2 7 10 7 5 1 0 3 9 0 1 -1 </pre>	<pre> !OK !OK !OK !OK !OK !OK !OK 6 7 !OK !OK </pre>
---	--

Infinite snakes and ladders

Time limit: 4760 ms

Memory limit: 264 MB

Snakes and ladders is a well known board game that goes back many centuries. Rules are simple, players start at the bottom left, roll the dice to move in a zig-zag path towards the top-left corner, use ladders to get higher, eaten by snakes to get back to their tail. Your task is to simulate the board game among players at a given board.



Example

We have a 4×4 board (positions 1 to 16) and two players that start outside the board at position 0 and win at position 17.

1	17	16	15	14	13
2		9	10	11	12
3		8	7	6	5
4	0	1	2	3	4
5					

Each cell can be identified through a plain XY axis coordinate system as follows:

1	(0,4)	(1,4)	(2,4)	(3,4)	(4,4)
2		(1,3)	(2,3)	(3,3)	(4,3)
3		(1,2)	(2,2)	(3,2)	(4,2)
4	(0,1)	(1,1)	(2,1)	(3,1)	(4,1)
5					

Note that in this 4×4 board, from (1, 1) up to (4, 4) are positions on the board while A (0, 1) is the starting point and B (0, 4) is the winning position. Both A and B are one cell to the left of (1, 1) and (1, 4) respectively.

We have a single **snake** and a single **ladder**:

- **Snake** with the head at position (4, 1) and the tail at position (2, 1).
- **Ladder** which starts at (4, 2) and ends at (2, 3).

Players always start at (0, 1) and roll two 6-sided dice, the sum is the number of position they move. The dice rolls are given in the input for our example: 1 4, 4 1, 3 1, 5 3, 5 6, 5 4, 1 4, 1 4, 1 3, 3 3.

1	Player	Dice	Before	Why	After
2	1	1+4 = 5	0 at (0,1)	dice	5 at (4,2)
3			5 at (4,2)	ladder	10 at (2,3)
4	2	4+1 = 5	0 at (0,1)	dice	5 at (4,2)
5			5 at (4,2)	ladder	10 at (2,3)
6	1	3+1 = 4	10 at (2,3)	dice	14 at (3,4)
7	2	5+3 = 8	10 at (2,3)	dice	18. This player won because they reached position 17.
8	1	5+6 = 11	14 at (3,4)	dice	25. This player won because they reached position 17.
9	-	5+4	Just rolling the dice, all players have won		
10	-	1+4	Just rolling the dice, all players have won		
11	-	1+4	Just rolling the dice, all players have won		
12	-	1+3	Just rolling the dice, all players have won		
13	-	3+3	Just rolling the dice, all players have won		
14					

Standard input

For the example above, input is given from standard input as follows:

```
1 4 // Board size, always an even number within [4, 10^6]
2 2 // P Number of players, integer within [2, 10]
3 1 // S Number of snakes, integer within [0, 10^5]
4 4 1 2 1 // S lines, for each snake head towards snake tail
5 1 // L Number of ladders, integer within [0, 10^5]
6 4 2 2 3 // L lines, for each ladder start towards ladder end
7 10 // K Number of roll dice pairs, integer in [1, 2 x 10^6]
8 1 4 // 1st roll of two 6-sided dice
9 4 1 // 2nd..
10 3 1
11 5 3
12 5 6
13 5 4
14 1 4
15 1 4
16 1 3
17 3 3 // K-th roll
18
```

Standard output

For each player, counting from player #1 towards player #P, print the player number followed by the word `winner` (if the player has won) or followed by the coordinates x, y of his final position. For our example, the output would be the following:

```
1 1 winner
2 2 winner
3
```

In case we had a different input and player 1 had reached cell 6 and player 2 had never rolled the dice (an acceptable situation), the output would be:

```
1 1 3 2
2 2 0 1
3
```

Constraints and notes

- When a player has already won, skip their turn and use their dice rolls for the players still in the game. If every player has already won, the remaining dice rolls are unused.
- **Ladders** always go up, **snakes** always go down. Never use a **ladder** to go down or a **snake** to go up.
- A position can not have both a **snake** head and a start from a **ladder**.
- If after following a **ladder** or a **snake**, a player ends up at a position with a **ladder** start or a **snake** head, they must follow it as well. However, there will be no infinite loops of **ladders** and **snakes**.

Input

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

Output

```
1 winner
2 winner
```

Lemons

Time limit: 680 ms

Memory limit: 264 MB

If life gives you lemon trees, make sure they have enough water!

Pantelis owns a small field of lemon trees on the foot of the Troodos mountain. He has chosen to grow his trees at this location mainly because Troodos mountain has a continuous source of fresh water for watering the trees. The water captured on the slopes, at high altitudes, flows under gravity via the stream and it is stored and distributed through an underground network of water pipes and water pumps.

Unfortunately, earlier this morning, Pantelis noticed that the flow of water from the water pump in his field is significantly reduced, and this will severely affect this season's crop of lemons. Watering lemon trees is tricky. Too little or too much water and the tree will die. You should never let a lemon tree dry out completely for more than a day. Pantelis suspects that one of the water pumps on the channel has malfunctioned causing the water shortage. He needs to immediately determine which pump malfunctioned, notify maintenance in order to get it fixed and re-establish the regular water flow to his trees. Since the water runs in underground pipes he has no way of knowing if the flow is reduced in the pipes. He needs to climb the mountain and methodically check each of the water pumps for shortage in water levels in order to figure out which one is damaged.

There are N water pumps evenly set apart along the network, with the first one (last in the flow line) being located at Pantelis's field. Pantelis knows that his pump is working properly. All the pumps ranging from the damaged pump down to Pantelis's pump are having water shortage. It takes Pantelis M minutes to get to the next pump on the mountain (uphill or downhill) and S minutes to check the pump for water shortage.

Pantelis needs to minimize the time needed to find the damaged water pump. Therefore, your program should find a plan to minimize the time to find the damaged water pump in the worst case scenario.

Standard input

The input consists of three space-separated integers N , M and S . N is the total number of water pumps on the network. Pantelis pump is number 1. M is the time it takes Pantelis to travel to the next pump of the network (uphill or downhill), and S is the number of minutes it takes to check if a water pump is having water shortage.

Standard output

The output consists of a single integer. The minimum worst-case time to find the damaged water pump.

Constraints and notes

- $2 \leq N \leq 10^4$
- $0 \leq M \leq 1\,000$
- $0 \leq S \leq 1\,000$

Input	Output	Explanation
4 1 2	7	<p>There are four water pumps on the channel. Pantelis's pump is number 1 and it has water shortage.</p> <p>Each of the pumps has $M = 1$ and $S = 2$. Pantelis needs one minute to reach the next pump in line and he needs two minutes to check each pump for water shortage. Pantelis checks pump 3 first. If it does not have water shortage, it means that pump 2 is the damaged one and he does not need to check water pump 4 (total time = 4). If water pump 3 has water shortage, then Pantelis will check water pump 4 next. If water pump 4 does not have water shortage it means that water pump 3 is the damaged pump, otherwise if water pump 4 has water shortage then pump 4 is the damaged one (total time = 7).</p>

Make Distinct

Time limit: 1580 ms

Memory limit: 264 MB

You are given an array of N integers, A . An operation involves taking a number from the array and either adding or subtracting 1 from it. What is the minimum number of operations to make A have only distinct elements.

Standard input

The first line contains an integer N .

The second line contains N elements, representing A .

Standard output

Print the answer on the first line.

Constraints and notes

- $1 \leq N \leq 2 \cdot 10^5$
- $1 \leq A_i \leq N$ for all valid i

Input	Output	Explanation
<pre>7 2 3 4 2 2 1 3</pre>	<pre>7</pre>	Transform into <code>2 4 5 -1 1 0 3</code> with a cost equal to <code>(0 + 1 + 1 + 3 + 1 + 1)</code> .
<pre>4 3 3 3 3</pre>	<pre>4</pre>	Transform into <code>5 4 3 2</code> .
<pre>5 5 4 4 5 4</pre>	<pre>4</pre>	Transform into <code>6 3 4 5 2</code> .

Telescope Scheduling

Time limit: *12080 ms*

Memory limit: *264 MB*

Joe and his friends want to observe the stars tonight using a unique telescope. Joe is an astronomy student and has a list of each of the stars that will be visible in the sky. Each star appears inside a certain time window with the possibility of multiple stars to overlap, so the group assigned a value to each star indicating the desirability of observing it.

The input consists on a list L of time intervals in which the stars will be available for observation. Each interval $i \in L$ consists of the following elements:

- a start time S_i after which the star will be available for observation;
- a finish time F_i after which the star will no longer be available;
- a positive integer D_i indicating the desirability to see the i -th star.

In order to satisfy the desirability of seeing the i -th star, the observations must be performed by the telescope for the entire time period from S_i to F_i (inclusive). Thus, two stars, i and j , are not simultaneously observable (i.e. they **conflict**) if the time interval $[S_i, F_i]$ intersects the time interval $[S_j, F_j]$. Given the list L of time intervals of availability of the stars in the sky, the optimization problem is to schedule the observations in a non-conflicting way so as to maximize the total desirability of the observations that are included in the schedule.

Standard input

The first line of the input contains a positive integer N indicating the number of stars.

Each of the following i lines, $1 \leq i \leq N$, indicates the start (S_i) and finish (F_i) times of each star together with the desirability D_i of seeing that star.

Standard output

Print the sum of the desirability of the stars included in the schedule on the first line.

Constraints and notes

- $1 \leq N \leq 10^4$
- $0 \leq S_i, F_i \leq 1\,000$
- $1 \leq D_i \leq 5\,000$

Input	Output	Explanation
<pre> 7 2 7 3 6 11 5 4 17 4 13 23 2 9 30 3 24 28 5 0 5 5 </pre>	17	<p>Star 2 is in conflict with star 1, 3, and 5 and it has a higher desirability (5), so star 1, 3 and 5 are not observed.</p> <p>All the desirability of the other stars adds up to 17 .</p>

Xplore

Time limit: 2480 ms

Memory limit: 264 MB

The IEEE Xplore Application Programming Interface (API) is an efficient data delivery vehicle for content indexing/discovery as well as text and data mining of IEEE metadata content of academic publications. Loading a database/repository using the content delivered by the IEEE API can be subsequently used to draw domain/subject relationships, data analytics, and various other use cases for researchers. To learn more about the IEEE Xplore API please visit developer.ieee.org and register for an API key. All participants of the IEEEExtreme 12.0 competition will have access to the IEEE API during and after the competition, for a limited period of time, to discover its research utility potential.

A useful metric commonly associated with academic publishing is the **h-index**. An author with an index of h has published h papers each of which has been cited in other papers at least h times.

For this challenge, write a program that reads a set of N entries from the Xplore database, in a JSON format, and prints ALL author names followed by their **h-index**. The authors should be ranked by **h-index** and by alphabetical order in case of an **h-index** tie.

Standard input

The input will consist of an integer N , followed by N lines with a single article entry in each line in a JSON format.

Each entry will follow a format described in the Xplore API website:
developer.ieee.org/docs/read/Metadata_API_responses

Standard output

Print the authors ranked by their **h-index** followed by a space and by the **h-index** itself. The authors should be ranked alphabetically if there are ties.

Constraints and notes

- $2 \leq N \leq 10000$
-

Input

```
10
{"authors": {"authors": [{"author_order": 1,"affiliation": "", "full_name": "Echo"},
{"authors": {"authors": [{"author_order": 1,"affiliation": "", "full_name": "Charlie"}
{"authors": {"authors": [{"author_order": 1,"affiliation": "", "full_name": "Echo"},
{"authors": {"authors": [{"author_order": 1,"affiliation": "", "full_name": "Charlie"}
{"authors": {"authors": [{"author_order": 1,"affiliation": "", "full_name": "Charlie"}
{"authors": {"authors": [{"author_order": 1,"affiliation": "", "full_name": "Charlie"}
{"authors": {"authors": [{"author_order": 1,"affiliation": "", "full_name": "Delta"}]}
{"authors": {"authors": [{"author_order": 1,"affiliation": "", "full_name": "Charlie"}
{"authors": {"authors": [{"author_order": 1,"affiliation": "", "full_name": "Delta"}},
{"authors": {"authors": [{"author_order": 1,"affiliation": "", "full_name": "Bravo"}},
```

Output

```
Charlie 5
Echo 4
Alfa 3
Bravo 3
Delta 3
```

Explanation

In this list, Charlie is the author of 7 papers: with `article_number` 2, 3, 4, 5, 6, 8, and 9. His papers have citation counts of 9, 4, 9, 5, 6, 9, and 4 respectively.

If we order his papers by citation count it will be: 9, 9, 9, 6, 5, 4, 4. Charlie's ***h-index*** is 5 Because he has 5 papers with at least 5 citations.

The same method is applied for Echo, Alfa, Bravo, and Delta. Because Alfa, Bravo, and Delta all have the same ***h-index*** they are listed alphabetically.

The Gold-diggers

Time limit: *2000 ms*

Memory limit: *264 MB*

Given a simple **connected graph** with vertices indexed from 0 to $N - 1$, N being the order of the graph, two players start at vertex 0 . We call the players **IEEE** and **Xtreme**.

There are 7 gold coins, located at vertices $N - 7, N - 6, \dots, N - 1$ (one per vertex).

The aim of the game is to catch as many gold coins as possible.

We call **move** for a player the change in location from the vertex he is currently at to one of its neighbors (i.e. if the player is at vertex u , it can go to any vertex v such that (u, v) is an edge in the graph).

The game is played as follows: alternatively and starting with **IEEE**, each player makes a **move**. If the vertex the player goes to contains a gold coin, it is immediately added to its collection and removed from the vertex. The two players can be at the same location at any time (and they start at the same vertex), but if that vertex initially contained a gold coin, only the first player to make the **move** collects it.

We call **strategy** for **IEEE** (respectively for **Xtreme**) a function that takes as input the current state of the game (position of the players and remaining gold coins) as well as the history of all **moves**, and associates a **move** for **IEEE** (respectively for **Xtreme**).

Thus, given one strategy for **IEEE** and one for **Xtreme** it is possible to compute how many gold coins are taken by **IEEE** during the game. Note that the game may be infinite.

We call **value** of a strategy for **IEEE** the **minimum** (i.e. provided that both players have acted in a way that maximized the gold coin collection, knowing that the strategy of the other player would similarly be to maximize the acquisition of gold coins) number of gold coins **IEEE** takes, considering any possible strategy for **Xtreme**.

We would like to know what is the **value** of the best strategy (the one with the maximum value) for **IEEE**.

Standard input

The input of your program is organized as follows:

- The first line contains only the order of the graph, N .
- The $i + 1^{\text{th}}$ line contains the list of neighbors of vertex i , separated by spaces, in no specific order.

0 1 2 3 4 5 6 7

Standard output

Print the answer on the first line.

Constraints and notes

- $8 \leq N \leq 20$
- (u, v) is an edge if and only if (v, u) is an edge for all valid u and v

Input	Output	Explanation																								
10 1 0 2 1 3 2 4 3 5 4 6 5 7 6 8 7 9 8	7	<p>This graph is in fact a straight line with the 10 vertices positioned linearly on it. Since the number of vertices is $N = 10$, the gold coins would be positioned on vertices 3, 4, 5, 6, 7, 8 and 9. As there is only a single way to go from vertex u to the vertex $u + 1$ (e.g. from 3 to 4) and the IEEE player always makes the first move, he would always come to each of the gold-containing vertices first (followed by the XTREME player on the next move) and hence the minimum number of gold coins the IEEE player would acquire would be 7. The exact game play would be as follows:</p> <table border="1"><thead><tr><th>#</th><th>Player</th><th>Move</th></tr></thead><tbody><tr><td>1</td><td>IEEE</td><td>0 → 1</td></tr><tr><td>2</td><td>XTREME</td><td>0 → 1</td></tr><tr><td>3</td><td>IEEE</td><td>1 → 2</td></tr><tr><td>4</td><td>XTREME</td><td>1 → 2</td></tr><tr><td>5</td><td>IEEE</td><td>2 → 3 (gets 1 coin)</td></tr><tr><td>6</td><td>XTREME</td><td>2 → 3</td></tr><tr><td>7</td><td>IEEE</td><td>3 → 4 (gets 1 coin)</td></tr></tbody></table>	#	Player	Move	1	IEEE	0 → 1	2	XTREME	0 → 1	3	IEEE	1 → 2	4	XTREME	1 → 2	5	IEEE	2 → 3 (gets 1 coin)	6	XTREME	2 → 3	7	IEEE	3 → 4 (gets 1 coin)
#	Player	Move																								
1	IEEE	0 → 1																								
2	XTREME	0 → 1																								
3	IEEE	1 → 2																								
4	XTREME	1 → 2																								
5	IEEE	2 → 3 (gets 1 coin)																								
6	XTREME	2 → 3																								
7	IEEE	3 → 4 (gets 1 coin)																								

8	XTREME	3 → 4
9	IEEE	4 → 5 (gets 1 coin)
10	XTREME	4 → 5
11	IEEE	5 → 6 (gets 1 coin)
12	XTREME	5 → 6
13	IEEE	6 → 7 (gets 1 coin)
14	XTREME	6 → 7
15	IEEE	7 → 8 (gets 1 coin)
16	XTREME	7 → 8
17	IEEE	8 → 9 (gets 1 coin)
18	XTREME	8 → 9

```

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

```

4

This graph is a **complete graph** with 10 vertices. As such, it is possible to move from any vertex to any vertex.

There are two cases: either **IEEE** decides to move each turn to a vertex that contains a gold coin, or at some point it goes to a vertex that does not contain a gold coin. In both cases, **XTREME** can decide to move systematically to a vertex that contains a gold coin. If the first case, and since **IEEE** plays first, he will catch exactly 4 gold coins. In the second case, he will catch at most 3 gold coins. We conclude that the value of the best strategy for **IEEE** is exactly 4.

In more details, a tight play would be:

#	Player	Move
1	IEEE	0 → 3 (gets 1 coin)
2	XTREME	0 → 4 (gets 1 coin)
3	IEEE	3 → 5 (gets 1 coin)
4	XTREME	4 → 6 (gets 1 coin)
5	IEEE	5 → 7 (gets 1 coin)
6	XTREME	6 → 8 gets (1 coin)
7	IEEE	7 → 9 (gets 1 coin)

Other tight plays could be obtained by renaming vertices 3 to 9

Tweedledee's Split Brackets

Time limit: 2480 ms

Memory limit: 264 MB

Tweedledee has a bracket sequence S with round and square brackets.

He begins an exercise with two empty sequences S_1 and S_2 . For each bracket in S , from left to right, Tweedledee decides to append that bracket to either S_1 or S_2 . He wants S_1 and S_2 to have equal length and both be valid bracket sequences. How can he split S into S_1 and S_2 ?

A valid bracket sequence is defined as follows:

- `()` and `[]` are valid bracket sequences;
- If S is a valid bracket sequence, then `(S)` and `[S]` are also valid bracket sequences;
- If Q and R are two valid bracket sequences, then the concatenation of them, $S = QR$, is also a valid bracket sequence.

Standard input

The input has a single line consisting of the bracket sequence S .

Standard output

For each bracket in s from left to right, output 1 to append the bracket to S_1 , or 2 to append the bracket to S_2 . Separate the numbers by single spaces. If there are multiple answers, output the lexicographically smallest sequence. If there is no solution to make the split, output the word `impossible`.

Constraints and notes

- $4 \leq |S| \leq 36$

Input	Output	Explanation
<code>([()])</code>	<code>1 1 1 2 1 2 2 2</code>	This answer splits the sequence into <code>([])</code> and <code>()</code> . Although <code>1 2 2 1 1 1 2 2</code> is also a valid split, <code>1 1 1 2 1 2 2 2</code> is lexicographically smaller.
<code>([])</code>	<code>1 2 1 2</code>	The only valid answer is to split into one pair of round brackets, and one pair of square brackets.
<code>(([])</code>	<code>impossible</code>	
<code>(() [])</code>	<code>impossible</code>	Although it is possible to split the sequence into two valid bracket sequences, their lengths cannot equal.
<code>([([]) [(])])</code>	<code>1 1 1 2 1 1 2 2 1 2 2 2</code>	

Tweedledum's Nested Brackets

Time limit: 1280 ms

Memory limit: 264 MB

The nested level of a bracket sequence S is defined as the number of unmatched left brackets that may exist in a prefix of S . For example, the nested levels of $((()))$, $(())()$, and $((())())$ are 3, 2, and 3 respectively.

Tweedledum has a string X consisting of a valid sequence of round brackets. In how many ways can you create a maximally nested bracket sequence by removing a continuous substring Y from X and re-inserting it into X at any location? The substring Y has to be a valid bracket sequence.

A valid bracket sequence is defined as follows:

- $()$ is a valid bracket sequence;
- If S is a valid bracket sequence, then (S) is also a valid bracket sequence;
- If Q and R are two valid bracket sequences, then the concatenation of them, $S = QR$, is also a valid bracket sequence.

Two ways are considered different if you either choose a different substring Y , or if you re-insert Y at a different position. Two equal substrings occurring at two different positions in X are considered two different choices of Y . Re-inserting Y at its original position counts if it is a maximally nested bracket sequence.

Standard input

The input has a single line consisting of the bracket sequence X .

Standard output

Output the number of ways in which you can create a maximally nested bracket sequence, by choosing and re-inserting a substring Y from X .

Constraints and notes

- $2 \leq |X| \leq 10^6$

Constraints and notes

- $2 \leq |X| \leq 10^6$

Input	Output	Explanation
<code>(())()</code>	2	<code>(())</code> on the left can be moved into <code>()</code> on the right, or vice versa.
<code>((()))</code>	3	<code>()</code> , <code>(())</code> , or <code>((()))</code> can all be re-inserted at their original positions.
<code>() () ()</code>	8	There are 6 ways to choose one pair of brackets and insert it into either of the other two pairs and there are 2 ways to choose two consecutive pairs of brackets and insert them into the remaining pair of brackets.

Friendly Sequences

Time limit: 1280 ms

Memory limit: 264 MB

Semar and Petruk are best friends and they love playing with integer sequences. Semar has a sequence that contains N integers, represented by $\{B_1, B_2, \dots, B_N\}$. Semar asks Petruk to count how many integer sequences $\{A_1, A_2, \dots, A_N\}$ have the following properties:

- $0 \leq A_1 \leq \dots \leq A_n$
- $A_1 \leq B_1$
- $A_1 + A_2 \leq B_2$
- $A_1 + A_2 + A_3 \leq B_3$
- ...
- $A_1 + \dots + A_N \leq B_N$

Your task is to help Petruk find the answer.

Standard input

The first line of the input contains integer N .

The second line of the input contains N integers representing the $\{B_1, B_2, \dots, B_N\}$ sequence, each integer is separated by one space.

Standard output

Print one integer representing the number of sequences with the given property.

As this number can be very large, output its value modulo $10^9 + 7$.

Constraints and notes

- $1 \leq N \leq 10^5$
- $0 \leq B_i \leq 2\,000$

Input	Output	Explanation
2 1 2	4	There are 4 possible sequences: {0, 0}, {0, 1}, {0, 2}, and {1, 1}.
4 13 51 30 73	39564	

Gotta ~ Catch 'Em All

Time limit: 1280 ms

Memory limit: 264 MB

Diglett is a ground-type Pokémon. Diglett digs regularly through the earth at a shallow depth, leaving perfectly tilled soil in its wake. Diglett lives in the forest, in caves under the earth, where it feeds on tree roots.

A greedy construction company is planning to build a tunnel near the Viridian city and they know that there are a lot of Digletts in the Viridian forest near the Viridian city. Instead of spending money on machines to dig a tunnel for the company, the company has decided to capture Digletts from the Viridian forest in order to use them for digging the tunnels so that they can reduce spending and increase profit margins. However, Digletts do not want to dig for a greedy construction company and want to be free.

Digletts can only be captured if they are on the surface and can escape by going into a hole. Whenever the Diglett senses a human presence, it goes into a hole. There are M holes and N Digletts. Initially, a hole can only accommodate a single Diglett. However, a Diglett can dig deeper in the hole in order to accommodate another Diglett. The time taken to dig deeper is T . A hole can accommodate two Digletts at most no matter how deep it is.

You are given the time required by each Diglett to travel from its current location to reach each hole. Your job is to find the minimum amount of time it will take before at least L Digletts are hiding in holes and are safe from the construction company.

Standard input

The first line contains the number of test cases K , followed by the data of each test case.

The first line of each test case contains 4 integers: M (number of holes), N (number of Digletts), L (minimum number of Digletts that need to go to holes) and T (time required by a Diglett to dig the hole deeper in order to hold another Diglett).

The next N lines contains M integers denoting time taken for each Diglett to reach each hole.

Standard output

Output a single integer denoting the minimum amount of time it will take before at least L Digletts are hiding in holes and are safe from the construction company.

Constraints and notes

- $1 \leq K \leq 10$
- $1 \leq M \leq 100$
- $1 \leq N \leq 100$
- $1 \leq L \leq \min(N, 2M)$
- $1 \leq T \leq 200$
- $1 \leq \text{Distance from Diglett to hole} \leq 100$

Input	Output	Explanation
<pre>1 3 1 1 100 100 100 50</pre>	50	<p>There is only one test case with 3 holes, 1 Diglett, 1 Diglett to save and $T = 100$ as cost to dig a hole deeper. The distances from Diglett to the three holes are 100, 100 and 50. The closest hole is 50 away. Hence, output is 50.</p>
<pre>2 3 3 2 50 10 20 50 25 30 40 100 60 30 3 4 3 10 10 11 100 11 12 100 15 100 100 100 100 100</pre>	25 20	<p>There are two test cases.</p> <p>For the first test case, there are 3 holes, 3 Diglett, 2 Diglett to save and $T = 50$ as the cost to dig a hole deeper. For the first Diglett, the distances from Diglett to the three holes are 10, 20 and 50. For the second Diglett, the distances from Diglett to three the holes are 25, 30 and 40. For the third Diglett, the distances from Diglett to the three holes are 100, 60 and 30. The first Diglett can go to the second hole and the second Diglett can go to the first hole. Hence, time required is 25.</p> <p>For the second test case, there are 3 holes, 4 Diglett, 3 Diglett to save and $T = 10$ as the cost to dig a hole deeper. For the first Diglett, the distances from Diglett to the three holes are 10, 11 and 100. For the second Diglett, the distances from Diglett to the three holes are 11, 12 and 100. For the third Diglett, the distances from Diglett to the three holes are 15, 100 and 100. For the fourth Diglett, the distances from Diglett to the four holes are 100, 100 and 100. The first Diglett goes to the first hole and starts digging so that the 3rd Diglett can also be accommodated. The second Diglett goes to the second hole. Hence, minimum time is 20.</p>

Product Rectangle

Time limit: 1280 ms

Memory limit: 264 MB

You are given an array A of N integers and an array B of M integers. Consider a matrix C of size $N \times M$ where $C_{i,j} = A_i \cdot B_j$. Find the maximum sum of a rectangle which contains at least one cell.

More formally, find $\max(\sum_{i=r_1}^{r_2} \sum_{j=c_1}^{c_2} C_{i,j} \mid 1 \leq r_1 \leq r_2 \leq N, 1 \leq c_1 \leq c_2 \leq M)$.

Standard input

The first line contains two integers N and M .

The second line contains N integers, representing A .

The third line contains M integers, representing B .

Standard output

Print the value of the sum on the first line.

Constraints and notes

- $1 \leq N, M \leq 2 \cdot 10^3$
- $-10^6 \leq A_i, B_j \leq 10^6$ for all valid i and j

Input	Output	Explanation
<pre>3 3 -1 3 5 -10 -3 20</pre>	<pre>160</pre>	$C = \begin{pmatrix} 10 & 3 & -20 \\ -30 & -9 & 60 \\ -50 & -15 & 100 \end{pmatrix}$ <p>We will choose $r_1 = 2, r_2 = 3, c_1 = 3, c_2 = 3$, which sums up to 160</p>

Tree Fun

Time limit: 1280 ms

Memory limit: 264 MB

You are given a tree with N nodes, each node has a score which is initially 0.

The input contains the structure of the tree as well as a list of M operations. Each operation identifies a pair of nodes (A, B) and a value K . With each operation the value of each node in the path between A and B is increased by K , including the starting and ending nodes.

After performing all the operations in the input, print the maximum score in the tree.

Standard Input

The first line of the input contains two integers: the number of nodes N , and the number of operations M .

The next $N - 1$ lines contain 2 integers U_i, V_i denoting an edge between U_i and V_i .

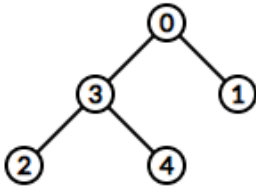
The following M lines contain three integers A_i, B_i, K_i representing the starting node, ending node, and score to add along the path.

Standard Output

A single integer representing the maximum score in the tree.

Constraints and notes

- $1 \leq N, M \leq 10^5$
 - $0 \leq U_i, V_i, A_i, B_i < N$
 - $1 \leq K_i \leq 1\,000$
-

Input	Output	Explanation
<pre> 5 2 0 1 0 3 2 3 3 4 4 1 4 2 4 3 </pre>	7	<p>$N = 5, M = 2$ This test case consists of a tree with 5 nodes in which we should do 2 operations.</p> <p>The list of connected nodes forms a tree like this:</p>  <pre> graph TD 0((0)) --- 3((3)) 0 --- 1((1)) 3 --- 2((2)) 3 --- 4((4)) </pre> <p>4 1 4 - updates the score from node 4 to node 1 by a value of 4 Path from 4 to 1 (4->3->0->1)</p> <p>2 4 3 - updates the score from node 2 to node 4 by a value of 3 Path from 2 to 4 (2->3->4)</p> <p>The maximum score is 7 (value of node 4 and 3)</p>

Factorial Zeros

Time limit: 1280 ms

Memory limit: 264 MB

Dahsser studies business administration in Lima but he is also an amateur programmer. During a particularly boring class he realized that the factorial of a number in a base B would end with a certain number of zeros, but that not all exact number of zeros was possible for a given base. He then tried to find an algorithm to print the minimum number whose factorial in base B ends with exactly N zeros.

Standard input

The first line contains an integer T , denoting the number of cases. The next T lines contain two separated integers B and N .

Standard output

For each case T , print the minimum number whose factorial in base B ends with exactly N zeros. In case there is no solution, print -1 .

Constraints and notes

- $1 \leq T \leq 20$
- $2 \leq B \leq 100$
- $1 \leq N \leq 10^{12}$

Input	Output	Explanation
2 3 2 10 5	6 -1	In the first case, 6 is the minimum number whose factorial in base 3 ends with exactly 2 zeros. $6! = 720$ (base 10), 222200 (base 3) In the second case there is no solution. There is no factorial in base 10 that ends with exactly 5 zeros.

Bit Soccer

Time limit: *680 ms*

Memory limit: *264 MB*

Peredo is a computer scientist who loves soccer. His favorite soccer player is Paolo Guerrero, one of the best Peruvian players, and his favorite team is the Brazilian national team.

He has a very large database of players with videos, photos, and many statistics related to their performance in hundreds of games. He uses his database to compute a binary **performance index** that tracks the players' abilities across 40 possible game metrics.

The **performance index** represents all possible soccer abilities of each player with a **0** for a lack of ability in a given game metric and a **1** for perfect ability, with no fractions in between **0** and **1**.

Based on these numbers, Peredo created a simulation game that takes the **performance indices** and combines multiple players to form a **team performance index**.

The **team performance index** is such that if a single player has a **1** in a given metric then the **team performance index** also has a **1** in that metric.

You are given a list of players in your **roster** represented by their **performance indices** in decimal format and your task is to combine a subset from your **roster** to form your **starting team** and to obtain a specific **team performance index**. There is no limit to the number of players that can form the **starting team**.

As an example, simplifying with just 4 game metrics, if we have two players on our **starting team** with **performance indices** **5** (**0101**) and **3** (**0011**) the resulting **team performance index** will be **7** (**0111**).

Standard input

The first line of the input contains an integer N denoting the number of player available in your **roster**. The second line contains N integers P_i , denoting the **performance indices** of each player. The third line contains an integer Q , denoting the number of queries, and each of the next Q lines contains an integer G that represents the goal **team performance index**.

Standard output

For each query, print **YES** if it is possible to select a **starting team** from the **roster** and obtain the **team performance index** G , otherwise print **NO**.

Constraints and notes

- $1 \leq N \leq 10^5$
- $1 \leq Q \leq 20$
- $1 \leq P_i, G \leq 2^{40} - 1$

Input	Output	Explanation
5 10 3 5 8 6 2 7 4	YES NO	In the first query, we can obtain 7 by selecting the players with performance indices 3 and 6. There is no way to obtain a team performance index of 4.

Barrett Reduction

Time limit: 680 ms

Memory limit: 264 MB

The C++ compiler optimizes the division by a constant D by multiplying your number by a constant A and then shifting the result B bits. This process is called **Barrett reduction**.

In this challenge we'll consider the division of an `unsigned 32 bit integer` by another `unsigned 32 bit integer`.

Given A and B , your task is to find the minimum possible value of D , for which the division is correct.

Note: it's guaranteed that for the given values A and B there exists an integer D such that for every number $0 \leq X < 2^{31}$, $\lfloor \frac{X}{D} \rfloor = \lfloor \frac{X \times A}{2^B} \rfloor$, where $\lfloor X \rfloor$ represents the floor (integer part) of the result.

E.g: $\lfloor 0.4 \rfloor = 0$, $\lfloor 6.8 \rfloor = 6$.

Note that the challenge only uses unsigned integers, so there are no problems regarding using floor on negative numbers.

Standard input

The first line contains two integers A and B .

Standard output

The first line should contain one integer D .

Constraints and notes

- $0 \leq A < 2^{32}$
- $0 \leq B \leq 63$
- $1 \leq D < 2^{31}$

Input	Output	Explanation
3435973837 35	10	Here are some examples of divisions. $\lfloor \frac{5}{10} \rfloor = 0$ $\lfloor \frac{5 \times 3435973837}{2^{35}} \rfloor = \lfloor \frac{17179869185}{2^{35}} \rfloor = 0$ $\lfloor \frac{17}{10} \rfloor = 1$ $\lfloor \frac{17 \times 3435973837}{2^{35}} \rfloor = \lfloor \frac{58411555229}{2^{35}} \rfloor = 1$ $\lfloor \frac{35}{10} \rfloor = 3$ $\lfloor \frac{35 \times 3435973837}{2^{35}} \rfloor = \lfloor \frac{120259084295}{2^{35}} \rfloor = 3$

Feedback Challenge

Time limit: *1280 ms*

Memory limit: *264 MB*

It has been almost 24 hours since you started this adventure and time is almost up!

There have been **rooted trees** and **fun trees**, **VERY large organizations** and **infinite ladders with snakes**! We are sure the **Troll Coder** made your life difficult, Bob surely did not make the life of **PIEEXMan** any easier. **Tweedledee and Tweedledum** had a very interesting obsession with **brackets**, maybe you can see their future research published on the **Xplore API**. There were **fake coins** and **gold-diggers**, the whole competition was full of rigors. While **Digletts** were running from that greedy construction company, did you **catch** that **magic spell** trick? Or were you busy **making distinctions** between the **product rectangle** and the **Barrett reduction**? You might have enjoyed playing a **bit-different** type of **soccer**, or maybe you prefer growing **lemons** and sell them in the market using the **barter system**. We hope you learned that **bears can sum**, and that **sequences can be friendly**, that is already more than the **factorial of zero**! Hopefully Xtreme has inspired you to **schedule your telescopes** and reach for the stars!

Your final task will be to share some feedback with us and ideas for improvements.

Let us know of your thoughts on how we can do better. As we said in the beginning, this event is FOR YOU and as such we want to learn from you to continue improving. Let us know what you feel went good or bad in this Xtreme and what you would do differently - aside from ensuring there are no downtime issue 😊 . Share any ideas on improvements with us and we promise we will read them all carefully and try to accommodate as many as we can in the next IEEEExtremes.

With your programming language of choice write a program that outputs a text with your feedback and submit it to the system. This task will not be scored and it is not mandatory.

Thank you for participating in the IEEEExtreme 12.0. We hope you had a lot of fun. See you next year.

Standard input

Nothing on standard input.

Standard output

Output a text with your feedback to us.

Constraints and notes

- Words \leq 200