ACM ICPC - Tehran Site Qualifications Round 1

Isfahan University of Technology

November 2019

Organizers:

A. Another Story of Moein-Ali

Moein-Ali likes sorting algorithms very much. He has studied quicksort, merge sort, radix sort, and many more.

A long time ago he has written a lock-free parallel string sorting program. It was a combination of burstsort and multi-key quicksort. To implement burstsort you need to build a tree of buckets. For each input string you walk through the tree and insert part of the string into the right bucket. When a bucket fills up, it ”bursts” and becomes a new subtree (with new buckets).

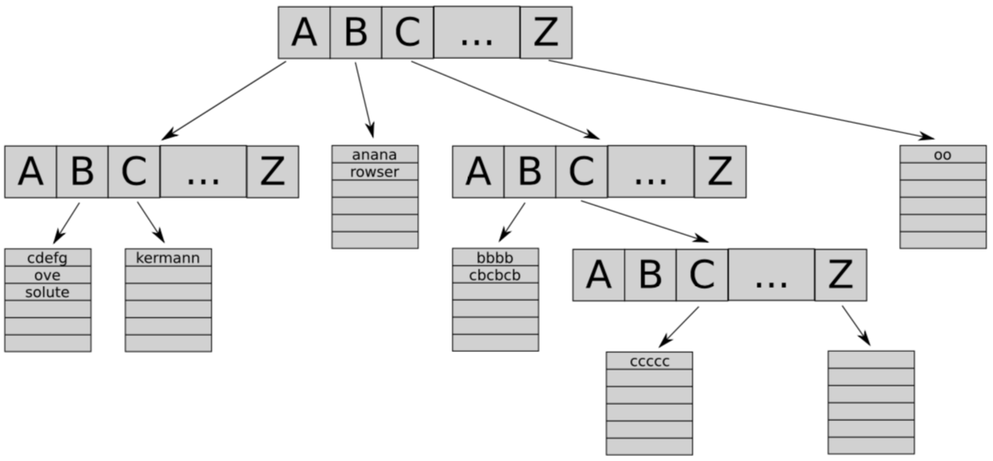


Figure 1: Burstsort data structure

Well, enough about the past. Today Moein-Ali is playing with sorting algorithms again. This time it's numbers. He has an idea for a new algorithm, “extreme sort”. It's extremely fast, performance levels are OVER MAMAD-ABAD. Before he tells anyone any details, he wants to make sure that it works correctly. Your task is to help him and verify that the so-called extreme property holds after the first phase of the algorithm. The *extreme property* is defined as $min(x_{i,j}) \geq 0$, where

$$x_{i,j} = \begin{cases} a_j - a_i & \text{for } 1 \leq i < j \leq N \\ 9001, & \text{otherwise} \end{cases}$$

Input

The first line contains a single integer N ($1 \leq N \leq 1024$). The second line contains N integers $a_1 a_2 ... a_N$ ($1 \leq ai \leq 1024$).

Output

Print one line of output containing “yes” if the extreme property holds for the given input, “no” otherwise.

Example

Standard Input	Standard Output

`2`
`1 2`
	`yes`

Standard Input	Standard Output

`4`
`2 1 3 4`
	`no`

B. Shuffle

You are listening to your music collection using the shuffle function to keep the music surprising. You assume that the shuffle algorithm of your music player makes a random permutation of the songs in the playlist and plays the songs in that order until all songs have been played. Then it reshuffles and starts playing the list again.

You have a history of the songs that have been played. However, your record of the history of played songs is not complete, as you started recording songs at a certain point in time and a number of songs might already have been played. From this history, you want to know at how many different points in the future the next reshuffle might occur.

A potential future reshuffle position is valid if it divides the recorded history into intervals of length s (the number of songs in the playlist) with the first and last interval possibly containing less than s songs and no interval contains a specific song more than once.

Input

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

- One line with two integers s and n ($1 \leq s, n \leq 100\ 000$): the number of different songs in the playlist and the number of songs in the recorded playlist history.
- One line with n space separated integers, $x_1, x_2, ..., x_n$ ($1 \leq x_i \leq s$): the recorded playlist history.

Output

Per testcase:

- One line with the number of future positions the next reshuffle can be at. If the history could not be generated by the above mentioned algorithm, output 0.

Example

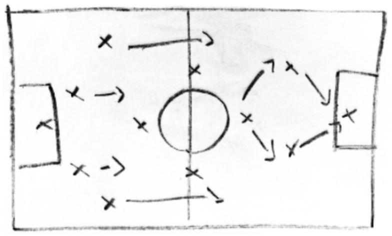
Standard Input	**Standard Output**

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

C. World Cup

World cup is ahead of us and we have to prepare ourselves. Coach **Marc Wilmots** wants to prepare Iran as well as possible. So he made up a strategy field plan for every player of the team. One plan describes a number of possible locations for the player on the field. Moreover, if Marc wants the player to be able to move from one location A to another location B then the plan specifies the ordered pair (A, B). He is sure that his team will win if the players run over the field from one location to another using only moves of the plan.

Marc tells every player to follow his plan and to start from a location that reaches every other location on the plan (by possibly multiple moves). However, it is quite difficult for some soccer players, simple minded as they are, to find a suitable starting location. Can you help every player to figure out the set of possible start locations?



Input

The first line gives the number of field plans. The input contains at most eleven field plans (what else?). Every plan starts with a line of two integers N and M, with $1 \leq N \leq 100\,000$ and $1 \leq M \leq 100\,000$, giving the number of locations and the number of moves. In the following M lines a plan specifies moves (A, B) by two white space separated integers $0 \leq A, B < N$. The plans are separated by a blank line.

Output

For every plan print out all possible starting locations, sorted increasingly and one per line. If there are no possible locations to start, print "Confused". Print a blank line after each plan output.

Example

Standard Input	**Standard Output**

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

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

	Confused
	```

D. Draughts

Draughts (or **checkers**) is a game played by two opponents, on opposite sides of a 10×10 board. The board squares are painted black and white, as on a classic chessboard. One player controls the dark, and the other the light pieces. The pieces can only occupy the black squares. The players make their moves alternately, each moving one of his own pieces.

The most interesting type of move is *capturing*: if a diagonally adjacent square contains an opponent's piece, it may be captured (and removed from the game) by jumping over it to the unoccupied square immediately beyond it. It is allowed to make several consecutive captures in one move, if they are all made with a single piece. It is also legal to capture by either forward or backward jumps.

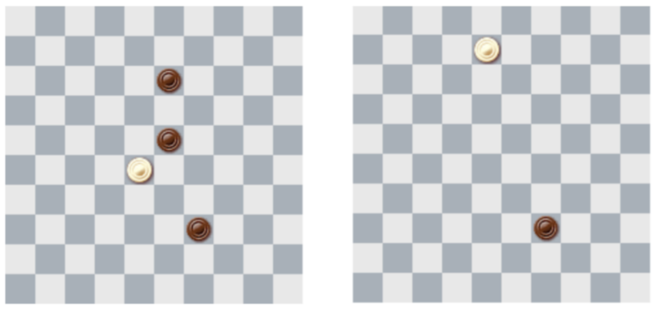


Figure 2: The board before and after a single move with two captures.

You are given a draughts position. It is the light player's turn. Compute the maximal possible number of dark pieces he can capture in his next move.

Input

The first line of input contains the number of test cases T. The descriptions of the test cases follow:

Each test case starts with an empty line. The following 10 lines of 10 characters each describe the board squares. The characters # and . denote empty black and white squares, `W` denotes a square with a light piece, `B` – a square with a dark piece.

Output

For each test case print a single line containing the maximal possible number of captures. If there is no legal move (for example, there are no light pieces on the board), simply output 0.

Example

Standard Input

```
 2

.#.#.#.#.#
#.#.#.#.#.
.#.#.B.#.#
#.#.#.#.#.
.#.#.B.#.#
#.#.W.#.#.
.#.#.#.#.#
#.#.#.B.#.
.#.#.#.#.#
#.#.#.#.#.

.#.#.#.#.#
#.#.#.#.#.
.#.#.B.#.#
#.B.#.B.#.
.#.#.B.#.#
#.B.W.#.#.
.#.B.B.#.#
#.#.#.#.#.
.#.B.B.#.#
#.#.#.#.#.
```

Standard Output

```
2
4
```

E. Isfahan Crisis

Once upon a time, there arose a huge discussion among people in Isfahan. The government wanted to introduce an identity card for all residents in Isfahan.

Most people accept to be rich, but they do not like to be measured. Therefore, the government allowed them to substitute the field “money amount” in their personal identity card with a field “relative money amount”. For producing the ID cards, people were being interviewed about their relative wealth. For some reason, the government suspects that at least one of the interviewed Isfahani people must have lied.

Can you help find out if the provided information proves the existence of at least one lying Isfahani?

Input

The input consists of:

- one line with an integer n ($1 \leq n \leq 100\,000$), where n is the number of statements;
- n lines describing the relations between the isfahanies. Each relation is described by:
  - one line with “$s_1 < s_2$” or “$s_1 > s_2$”, telling whether Isfahani s_1 is richer or poorer than Isfahani s_2. s_1 and s_2 are two different Isfahani names.

A Isfahani name consists of at most 20 letters from “A” to “Z” and “a” to “z”. A Isfahani name does not contain spaces. The number of Isfahanies does not exceed 10 000.

Output

Output “impossible” if the statements are not consistent, otherwise output “possible”.

Example

Standard Input	Standard Output

`3`
`Mehran > Alireza`
`Alireza > Mamad`
`Mehran < Mamad`
	`impossible`

Standard Input	Standard Output

`3`
`Mehran > Alireza`
`Alireza > Mamad`
`Mehran > Mamad`
	`possible`

F. Rick vs. Morty

The latest reality show has hit the TV: “Rick vs. Morty”. In this show, a bunch of Ricks and Mortys compete for the very prestigious BEST ADVENTURER EVER title. In each episode, the Ricks and Mortys get to show themselves off, after which the viewers vote on which ones should stay and which should be forced to leave the show.

Each viewer gets to cast a vote on two things: one person which should be kept on the show, and one person which should be thrown out. Also, based on the universal fact that everyone is either a Rick lover (i.e. a Morty hater) or a Morty lover (i.e. a Rick hater), it has been decided that each vote must name exactly one Rick and exactly one Morty.

Ingenious as they are, the producers have decided to use an advancement procedure which guarantees that as many viewers as possible will continue watching the show: the Ricks and Mortys that get to stay will be chosen so as to maximize the number of viewers who get both their opinions satisfied. Write a program to calculate this maximum number of viewers.

Input

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

- One line with three integers r, m, v ($1 \le r, m \le 100$ and $0 \le v \le 500$): the number of Ricks, Mortys, and voters.
- v lines with two person identifiers each. The first is the person that this voter wants to keep, the second is the person that this voter wants to throw out. A person identifier starts with one of the characters ‘`R`’ or ‘`M`’, indicating whether the person is a Rick or Morty, respectively. The remaining part of the identifier is an integer giving the number of the person (between 1 and R for Ricks, and between 1 and m for Mortys). So for instance, “`M42`” indicates Morty number 42.

Output

Per testcase:

- One line with the maximum possible number of satisfied voters for the show.

Example

Standard Input	**Standard Output**

`2`
`1 1 2`
`R1 M1`
`M1 R1`
`1 2 4`
`R1 M1`
`R1 M1`
`R1 M2`
`M2 R1`
	`1`
	`3`

G. Pizza Voting

Tahereh is training for a programming contest with her team members Maryam and Zahra. After some hours of hard training she wants to have a break and eat pizza. She decided to order a big pizza for all three of them. But she has to choose the kind of pizza she want to eat.

Tahereh knows her favorite kind. But Maryam and Zahra have other constraints: Maryam is on a diet so she wants a pizza with less calories as possible. Zahra is just mean to Maryam so he votes for as much calories as possible.

Tahereh decides to vote on which kind of pizza she order. As voting for one pizza wouldn't lead anywhere, she decides to use a veto voting. So everyone of them veto on pizza in a round robin manner. First Maryam vetos one pizza, then Zahra vetos one, at last Tahereh is allowed to veto. Then Maryam has the next veto again, then Zahra etc. until only one pizza is left.

Reminder: Maryam will always veto the pizza with the most calories. Zahra vetos always the pizza with least calories. Tahereh tries to be clever in such a way that her favorite pizza is the remaining pizza.

Input

The input starts with the number of pizzas n ($1 \leq n \leq 100\ 000$) and the index of Tahere's favorite pizza p ($1 \leq p \leq n$) (1-indexed). Then follow the description of the n pizzas, each given in one line. The description consists of one integer c ($0 \leq c \leq 1\ 000\ 000$) giving the calories followed by a single word w giving the name of the pizza (up to 100 characters). The pizzas are ordered from low calories to high calories and the number of calories is unique for every pizza.

Output

Output one line. "`YES`" if Tahereh can vote in a way, such that her pizza will be selected. "NO" if she is not able to influence the vote in a way that her pizza will be selected.

Example

Standard Input	**Standard Output**

`5 2`
`500 Margherita`
`600 Salami`
`700 Hawai`
`800 Speciale`
`900 Doener`
	`YES`

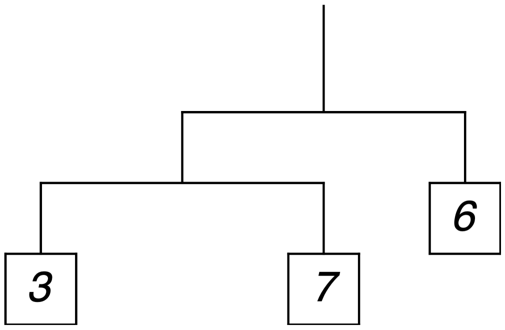
Standard Input	**Standard Output**

`5 4`
`500 Margherita`
`600 Salami`
`700 Hawai`
`800 Speciale`
`900 Doener`
	`NO`

H. Equilibrium Mobile

A mobile is a type of kinetic sculpture constructed to take advantage of the principle of equilibrium. It consists of a number of rods, from which weighted objects or further rods hang. The objects hanging from the rods balance each other, so that the rods remain more or less horizontal. Each rod hangs from only one string, which gives it freedom to rotate about the string.

We consider mobiles where each rod is attached to its string exactly in the middle, as in the figure underneath. You are given such a configuration, but the weights on the ends are chosen incorrectly, so that the mobile is not in equilibrium. Since that's not aesthetically pleasing, you decide to change some of the weights.



What is the minimum number of weights that you must change in order to bring the mobile to equilibrium? You may substitute any weight by any (possibly non-integer) weight. For the mobile shown in the figure, equilibrium can be reached by changing the middle weight from 7 to 3, so only 1 weight needs to changed.

Input

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

- One line with the structure of the mobile, which is a recursively defined expression of the form:

  `<expr> ::= <weight> | [ <expr> , <expr> ]`

  with `<weight>` a positive integer smaller than 10^9 indicating a weight and `[<expr>, <expr>]` indicating a rod with the two expressions at the ends of the rod. The total number of rods in the chain from a weight to the top of the mobile will be at most 16.

Output

Per testcase:

- One line with the minimum number of weights that have to be changed.

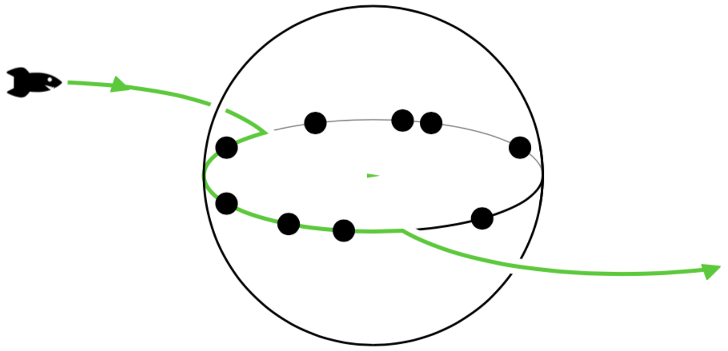
Example

Standard Input	Standard Output
3 [[3,7],6] 40 [[2,3],[4,5]]	1 0 3

I. Equator

In a galaxy far away, the planet Equator is under attack! The evil gang Interstellar Union of Tuxes is planning robberies in Equator's cities. Your help is needed! In order to complete your training for becoming a lord of the dark side you should help them deciding which cities to rob.

As the name says, the desert planet Equator only can be inhabited on its equator. So the gang lands there at some point and travels into some direction robbing all cities on their way until leaving the planet again.



But what is still open for them is to decide where to land, which direction to take, and when to leave. Maybe they shouldn't even enter the planet at all? They do not consider costs for traveling or for running their ship, those are peanuts compared to the money made by robbery!

The cities differ in value: some are richer, some are poorer, some have better safety functions. So the gang assigned expected profits or losses to the cities. Help them deciding where to begin and where to end their robbery to maximize the money in total when robbing every city in between.

Input

The input starts with the number of test cases $T \leq 30$. Each test case starts a new line containing the number of cities $1 \leq n \leq 1\,000\,000$. In the same line n integers c_i follow. Each c_i $(0 \leq i < n, -1\,000 \leq c_i \leq +1\,000)$ describes the money obtained when robbing city i, a negative c_i describes the amount of money they would lose.

Output

For each test case print one integer describing the maximum money they can make in total.

Example

Standard Input	Standard Output

`3`
`3 1 2 3`
`8 4 5 -1 -1 1 -1 -1 5`
`2 -1 -1`
	`6`
	`14`
	`0`

J. No One Loves Sheikhbahaei!

IUT has a super computer called Sheikhbahaei as you may know. It has 458 752 CPU cores (who knows?!). Since it has got quite old and currently almost no one uses it, and also it draws a lot of energy, we want to reduce the energy consumption by underclocking the unused cores. The cluster scheduling algorithm which is in charge of distributing jobs over the nodes and cores of a cluster will issue the following speedstepping commands:

- `change X S` changes the frequency of core `X` by `S` steps
- `groupchange A B S` changes the frequency of every core in range `[A,B]` by `S` steps
- `state X` returns the current state of core `X`

To be safe for the future, your program should be able to handle 4 587 520 cores. The initial frequency for each core is 0.

Input

The input contains a single test case. It starts with a line containing three integers C, N, and O, where C is the number of cores ($1 \leq C \leq 4\ 587\ 520$) to manage, N is the number of frequency steps for each core ($1 \leq N \leq 10\ 000$) and O is the number of operations in the test program ($1 \leq O \leq 50\ 000$). Then follow O lines, each containing one command as described above. X, A and B are 0-based IDs of the cores ($0 \leq A, B, X < C; A \leq B$). S is an integer number of steps, possibly negative ($-N \leq S \leq +N$).
Both, the `change` and the `groupchange` command will increase (or decrease) in single steps and stop as soon as **one** core in the group reaches the minimal (0) or maximal frequency (N).

Output

Output one line for every operation in the input. For `change` and `groupchange` print the changed number of steps, for `state` print the current state.

Example

Standard Input	Standard Output
10 10 5 state 0 groupchange 2 9 7 state 9 groupchange 0 2 10 change 0 -5	0 7 7 3 -3

Standard Input	Standard Output
4587520 10000 5 groupchange 0 4587010 9950 groupchange 23 4587000 42 groupchange 4710 4587001 -1000 state 1234560 groupchange 6666 3060660 10000	9950 42 -1000 8992 1008

K. Justified Jungle

As you probably know, a *tree* is a graph consisting of n nodes and $n-1$ undirected edges in which any two nodes are connected by exactly one path. A forest is a graph consisting of one or more trees. In other words, a graph is a forest if every connected component is a tree. A forest is justified if all connected components have the same number of nodes.

Given a tree G consisting of n nodes, find all positive integers k such that a justified forest can be obtained by erasing exactly k edges from G. Note that erasing an edge never erases any nodes. In particular when we erase all $n-1$ edges from G, we obtain a justified forest consisting of n one-node components.

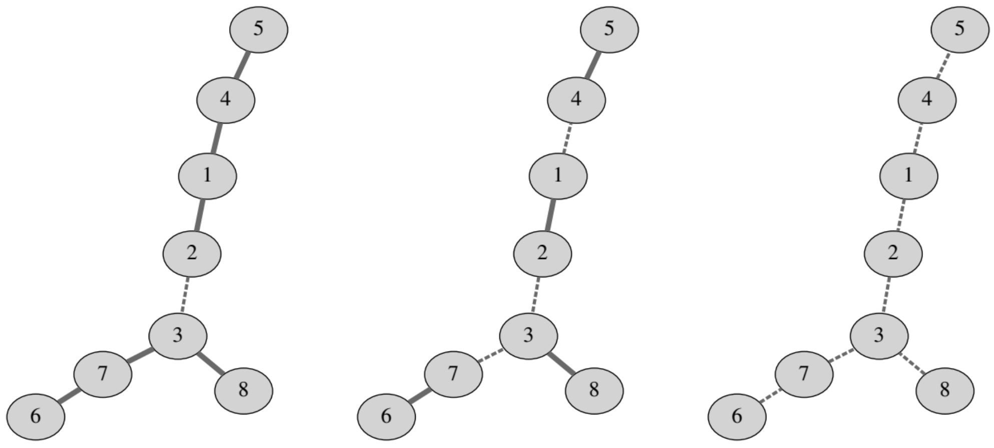


Figure 3: Figures depict justified forests obtained by erasing 1, 3 and 7 edges from the tree in the example input.

Input

The first line contains an integer n ($2 \le n \le 1\ 000\ 000$) — the number of nodes in G. The $k-th$ of the following $n-1$ lines contains two different integers a_k and b_k ($1 \le a_k, b_k \le n$) — the endpoints of the k-th edge.

Output

The first line should contain all wanted integers k, in increasing order.

Example

Standard Input	Standard Output

8
1 2
2 3
1 4
4 5
6 7
8 3
7 3
	1 3 7