MCPC 2025

Jackson & Parsa

Division B


A: *Tadpole Time*


B: *Rock Piles*


C: *Tickets*


D: *Treasures*


E: *The One that Got Away*


F: *Cats and Mouse*


G: *Magic and Merch*


H: *Greedy Cat*


I: *GiDi Up*


J: *Graduation*


K: *Birthday*

October 4, 2025

This page is intentionally left blank

A. Tadpole Time

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

In his travels Ali comes across a knot of toads, residing in a nearby pond.

The toads had recently given birth to a large school of tadpoles, who in their larval stage, will stay arranged in a single line to promote equal access to food and problem solvability. In this community, there are three types of toads (and therefore tadpoles):

1. A-type tadpoles are aggressive
2. B-type tadpoles are friendly
3. C-type tadpoles are shy

In fact, tadpoles are so fitting with this delineation that the following rules have been observed:

1. **Squish Rule**: If there are two A-type tadpoles which surround a group of entirely C-type tadpoles, then the C-type tadpoles will all be killed. (`ACCCCA -> AA`)
2. **Move Rule**: If there are two B-type tadpoles adjacent to each other, and a C-type tadpole immediately left of them, and an A-type tadpole immediately left of the C-type tadpole, then the C-type tadpole moves in-between the two B-types. (`ACBB -> ABCB`)
3. **Party Rule**: If there are 5 tadpoles in the following formation: `ABCBA`, then all 5 tadpoles become A-type tadpoles. (`ABCBA -> AAAAA`)

The tadpoles will only grow into toads once none of these rules can be applied to the line of tadpoles. It can be proven that regardless of the order of application of these rules, you will end up with the same final line.

If Ali can correctly predict the resulting line of tadpoles, Ali can help the toads plan for the future and he will gain their trust. Can you help him?

Input

Input will begin with a single integer n ($1 \leq n \leq 10^5$), denoting the number of tadpoles in the school.

A single line will follow, containing n characters `A`, `B` or `C` representing the initial line of tadpoles.

Output

Output should begin with a single integer m, representing the final length of the tadpole line.

After this, print a single line, containing m characters from `A`, `B` or `C` representing the final tadpole line.

Examples

standard input	standard output
20 ACCCACCBBCBBACBBACCA	15 AACCBBCBBAAAAAA

For the example input ACCCACCBBCBBACBBACCA, we could first apply the Squish rule to any collections of Cs encased by As:

```
ACCCACCBBCBBACBBACCA
=====           ====
AACCBBCBBACBBAA
```

Next, our only option is to apply the Move rule to any occurrences of ACBB:

```
AACCBBCBBACBBAA
     ====
AACCBBCBBABCBAA
```

Next, we can use the Party rule on any occurrences of ABCBA:

```
AACCBBCBBABCBAA
     =====
AACCBBCBBAAAAAA
```

After this, no more rules can be applied.

B. Rock Pile

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Ali decides to take residence among the toad civilization for a while. This understandably takes some adjusting, and one of the ways this presents itself is in their currency.

Toads use rocks as currency to trade for goods and services, like toad-school or toad-restaurants. Luckily for Ali, he already had r rocks in his pocket before arriving in the toad civilization.

Unfortunately for the toads, Ali knows how to break rocks, which allows him to turn few rocks into many.

So for Ali, there are two ways to increase his rock collection:

1. He can work a job i, earning him b_i rocks.
2. He can break all of his available rocks in two, doubling the amount of rocks in his collection.

However, this has three stipulations:

1. The job market in toad civilization is weak, and so for any job i you actually have to pay upfront a_i rocks to secure the job. (As the saying goes, you've got to spend rocks to make rocks.)
2. For a similar reason, you can only complete job i at most c_i times (each time, you need to pay a_i rocks).
3. Because Ali does not want to inflate the rock currency too much, he will only allow himself to break all of his rocks k times.

Within these stipulations, Ali wants to end up with the most rocks possible.

Input

Input will begin with 3 space-separated integers:

- r, the initial amount of rocks Ali has ($0 \leq r \leq 100$),
- n, the number of jobs available to Ali ($0 \leq n \leq 10^5$),
- k, the number of times Ali is allowed to break all of his rocks ($0 \leq k \leq 20$).

n lines will then follow, each containing 3 space-separated integers:

- a_i, the upfront cost Ali has to pay to complete this job ($0 \leq a_i \leq 10^9$)
- b_i, the rocks Ali will receive for completing this job ($0 \leq b_i \leq 10^9$)
- c_i, the number of times Ali can complete this job ($0 \leq c_i \leq 10^9$)

It is guaranteed that the sum of $(b_i - a_i) \times c_i$ over all jobs will not exceed 10^9.

Output

Your output should be a single integer, representing the largest amount of rocks that Ali could ever attain.

Examples

standard input	standard output
4 4 4 7 10 3 1000000 10000000 100 20 50 2 10 4 1000	376

One possible way to amass a rock pile would be like so:

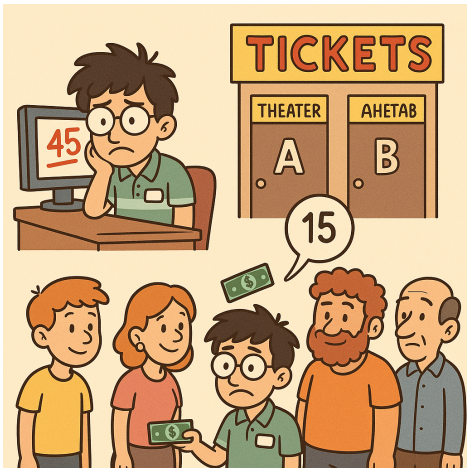
1. Currently, all jobs require more upfront rocks than Ali owns, so for now his only option is to double his rocks. Let's do this twice. We go from 4 to 16 rocks.
2. Now, we have access to 2 jobs, one costing 7 rocks and providing 10, and another costing 10 rocks and providing 4. Let's work the first job 3 times. We then end up with 25 rocks.
3. Let's double our rocks a third time, taking us to 50 rocks.
4. We now might work the 3rd job (20 rocks for 50 rocks) just once, leaving us at 80 rocks.
5. We can now use our last rock-doubling, to net us 160 rocks.
6. We can now work the 3rd job a second time, getting us to 190 rocks.

That is one possible way, but it is not the optimal way. The optimal way would net us '376' rocks in total.

standard input	standard output
5 4 0 0 10 1 10 20 1 40 50 5 20 30 2	95

C. Tickets

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes



After receiving his FIT1054 mark for assignment 1, Parsa decided to retire from competitive programming and pursue his true passion: selling cinema tickets.

Unfortunately, without a strong résumé, he could only find work at a very small cinema with two theatres. Each theatre can accommodate at most B people.

Every evening, n people line up to buy tickets. However, they have a peculiar habit: each person wants to watch the same movie as the person directly in front of them in the queue. The only way to make a person choose the other movie is to pay them exactly the amount of money they have in their pocket, at which point all subsequent people in line also change their mind.

The amount of money the i-th person has in their pocket is denoted by a_i. Parsa's boss has instructed him to sell tickets under the following conditions:

1. No theatre should contain more than B people.
2. Parsa should spend the minimum possible amount of money.

Since Parsa is traumatised by anything that reminds him of competitive programming, he has asked you to determine the minimum amount of money he must spend.

It is guaranteed that there exists a way to assign tickets such that each theatre has at most B people.

Input

The first line of input contains two integers:

- n ($1 \le n \le 3000$) – the number of people in the queue.
- B ($n \le 2B$) – the capacity of each theatre.

The second line contains n integers, where the i-th integer represents a_i, the amount of money the i-th person has in their pocket.

- a_i ($0 \le a_i \le 10^9$).

Output

Print a single integer – the minimum amount of money Parsa must spend.

Examples

standard input	standard output
1 1 0	0
2 1 1000 50	50
3 2 50 10 1000	10
6 3 0 1 2 3 4 5	3

In example 3, we have three patrons, and need to assign them so that no theatre has more than 2 people. The optimal way to do this is to pay only the second person to change their mind, then one theatre will have patron #1, while the other theatre will have patrons #2 and #3.

In example 4, we can simply put three patrons in each theatre by paying just patron #4.

D. Treasures

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Jackson has secretly buried treasures all over the Monash campus. The campus can be represented as an $n \times m$ grid, and there are k treasures hidden in it. The i-th treasure is buried at row x_i and column y_i of the grid.

Since the treasures are meant to support MAPS, Jackson has asked Eric, the treasurer of MAPS, to ensure they are divided fairly. To do this, Eric must choose one horizontal line and one vertical line across the grid lines (see the sample), splitting the campus into four regions. The division is considered fair only if each of the four regions contains the same number of treasures.

Eric, however, is exhausted from weeks of administrative work and doesn't want to check all possibilities by hand. That's where you come in: given the layout of the campus and the locations of the treasures, determine whether such a fair division is possible.

Input

The first line of input contains one integer:

- t ($1 \le t \le 1000$) – the number of scenarios.

For each scenario:

- The first line contains three integers n, m, k with $(2 \le n, m \le 10^9)$ and $(1 \le k \le 10^5)$, denoting the grid dimensions and the number of treasures.
- Each of the next k lines contains two integers x_i, y_i where $(1 \le x_i \le n)$ and $(1 \le y_i \le m)$ giving the row and column of the i-th treasure (1-indexed).

It is guaranteed that the sum of k across all scenarios does not exceed 10^5.

Output

For each scenario, print **"YES"** if a valid division exists; otherwise, print **"NO"**.

Examples

standard input	standard output
4 2 3 4 1 1 1 3 2 1 2 3 2 2 4 1 1 1 1 2 2 2 2 3 3 4 1 2 2 1 2 3 3 2 1000000000 1000000000 1 6 8	YES NO NO NO

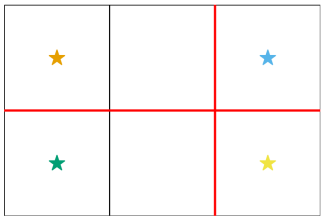


Figure 1: The grid in our first scenario can be divided as shown.

E. The One That Got Away

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	256 megabytes

The world of cuboids — strong, rectangular prisms that gave shape and order to the land — was under attack. In a cunning move, the wicked Curves struck and stole one corner from every cuboid, leaving them broken and unstable. A cuboid without all eight corners could not exist properly, and the entire land began to crumble.

In their despair, the cuboids turned to Alex, a hero known for his wisdom and precision. They placed before him the coordinates of seven corners of the damaged cuboids. *"From these seven points,"* they pleaded, *"find the missing one. Only then will we be whole again."*

Alex studied the problem carefully. He knew that in every such case:

- The shape in question is a rectangular prism (cuboid).
- Exactly one unique point exists that can serve as the missing eighth corner.
- When this point is found, the cuboid's volume is guaranteed to be positive, ensuring the block stands solidly in space.
- The cuboid's edges do not need to be parallel to the coordinate axes — they may be tilted and rotated — yet the structure remains a true cuboid.

Armed with this knowledge, Alex set out to recover the lost corner. If he succeeded, the cuboid would be restored to its full form, and the balance of the geometric world would be saved.

Input

The first line of input contains one integer:

- t ($1 \le t \le 40{,}000$) – the number of cuboids.

The description of the i-th cuboid follows, consisting of the spatial coordinates of seven of its vertices, each given on a separate line. For every coordinate (x_i, y_i, z_i) we know that:

- x_i, y_i, z_i ($0 \le x_i, y_i, z_i \le 10^8$).

Output

For each cuboid, output the spatial coordinates of its eighth vertex in a single line, following the format shown in the sample output.

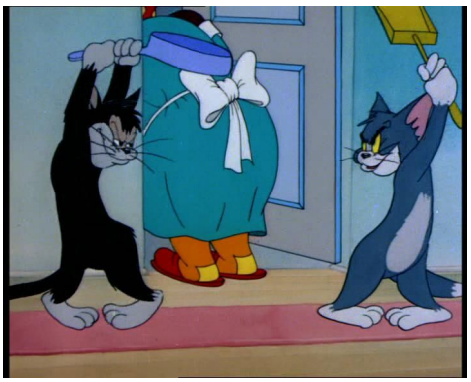
It is guaranteed that there exists exactly one eighth vertex with integer coordinates such that the cuboid has a positive volume.

Examples

standard input	standard output
2 0 0 1 0 1 0 0 1 1 1 0 0 1 0 1 1 1 0 1 1 1 6 16 12 11 12 13 5 14 9 11 11 9 10 10 10 6 15 8 12 13 12	0 0 0 7 17 11

F. Cats and Mouse

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	512 megabytes



After a hard fought battle, Ali now finds himself against a pair of undefeatable foes - two human sized cats.

These cats are inseparable, making it impossible for Ali to face either of them in one-on-one combat. However, their inseparability is also their weakness, for the cats need to always be within earshot of each other.

In particular, the battlefield Ali and the cats are battling on can be represented by a collection of positions, connected by tunnels. The outcome of the battle is determined in the following manner:

1. The two cats first get to pick the single location where they will start.
2. Ali then picks his start location. To keep it fair, this location must be reachable from the cat's starting location using the tunnels.
3. Now the foes take turns, repeatedly:
   (a) One of the cats may move to any location, with the restriction that their new location must be at the same position, or adjacent via a tunnel, to the stationary cat.
   (b) Ali can then move to an adjacent location by using a tunnel, or stand still.
   (c) If after he moves/stays still, he is in the same location as a cat, then he is defeated.
4. If Ali can evade the cats indefinitely (he has a strategy that means the cats will never catch Ali), then the cats are defeated due to exhaustion.

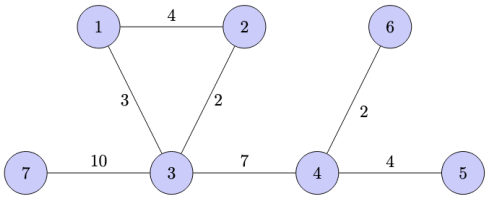
However, Ali was cunning, and knew this day would come. In fact, the tunnels in the battlefield are not natural - these were his own creation!

Digging tunnels takes energy however, and Ali wants to minimise his energy exerted while still guaranteeing victory.

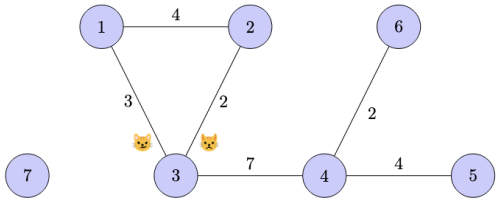
Your task is, given a number of locations, and the possible tunnels between them, find the cheapest way to ensure that Ali can always defeat the cats, if it is possible.

Explanation

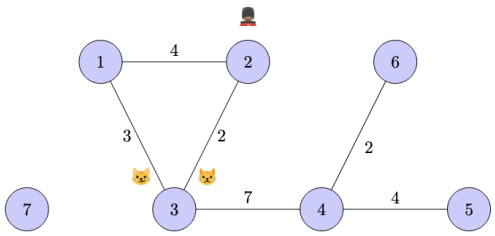
Let's play the game, step by step.
Suppose the battlefield looked like this, and Ali dug all tunnels except 3-7:



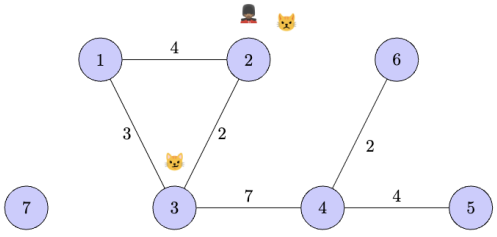
First, the cats get to pick their starting position.
Let's say they chose position 3:



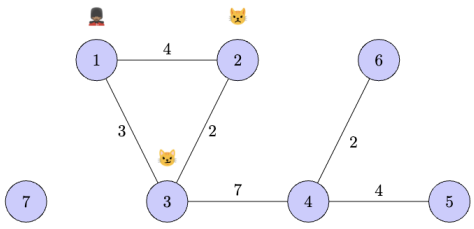
Next, Ali can pick any location to start which the cats can reach (so anywhere but 7).
Let's start at 2:



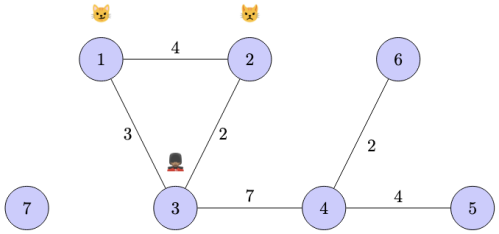
Now, we go into the repetitive part of the game.
Let's suppose that the pouting cat went to position 2:



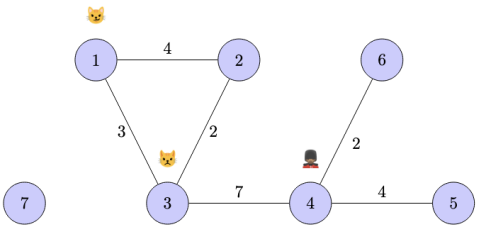
Now, Ali needs to pick an adjacent location not occupied by the cats.
So he can move to position 1:



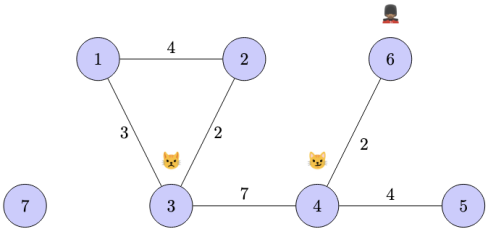
The smirking cat then moves to position 1, allowing Ali to move into position 3:



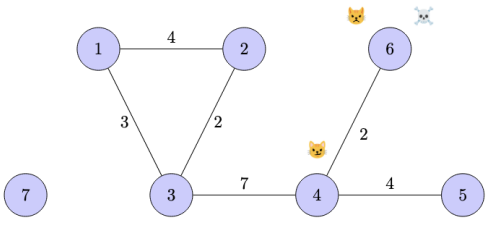
Next, the pouting cat moves back to position 3, and Ali then needs to move to either position 2 or 4.
He picks 4:



Smirking cat then moves to position 4, and Ali moves to position 6:



Now the pouting cat moves to position 6, and Ali has no move that doesn't land him in the same spot as a cat.
Ali is defeated:



Input

Input will begin with two integers n ($1 \leq n \leq 10^5$), the number of locations, and m ($0 \leq m \leq 10^5$), the number of possible tunnels.

m lines will follow, each representing a possible tunnel that Ali could dig. This line contains 3 integers:

- a ($1 \leq a \leq n$), the location on one side of the tunnel,
- b ($1 \leq b \leq n$), the location on the other side of the tunnel,
- c ($1 \leq c \leq 10^9$), the energy cost required to dig the tunnel.

Output

If it is impossible for Ali to guarantee victory, then simply print 'NO'.

If it is possible for Ali to guarantee victory, then find the collection of tunnels whose total energy cost is minimised while still guaranteeing victory.

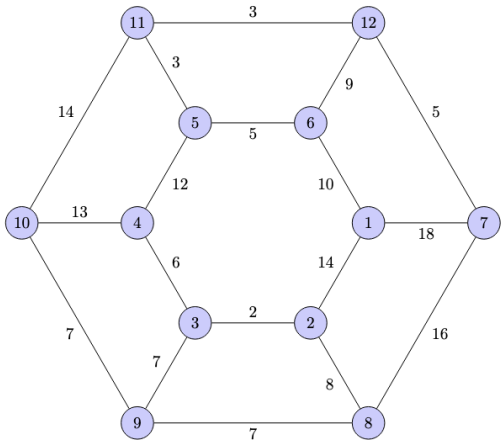
Start with a single line 'YES x' where x is the number of tunnels used.

x lines should then follow, each containing 3 integers 'a b c' representing the tunnel used (matching the input the tunnel was specified with). If there are multiple cheapest ways to dig tunnels, then choosing any valid selection will be accepted.

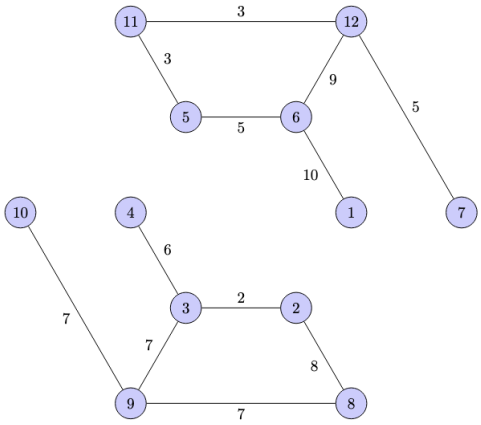
Examples

standard input	standard output
12 18 1 7 18 8 9 7 3 9 7 6 1 10 2 3 2 4 10 13 2 8 8 5 6 5 7 8 16 6 12 9 1 2 14 11 12 3 5 11 3 3 4 6 4 5 12 10 11 14 9 10 7 12 7 5	YES 12 3 9 7 6 1 10 2 3 2 9 10 7 3 4 6 8 9 7 5 6 5 6 12 9 12 7 5 11 12 3 2 8 8 5 11 3

This example is represented by the following diagram:

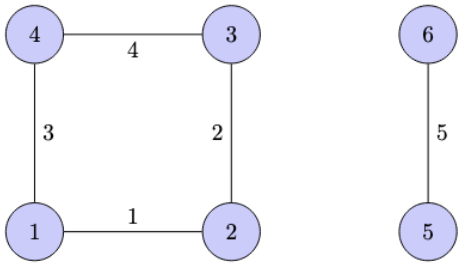


This can be made winnable for Ali by digging the following tunnels:



standard input	standard output
6 5 1 2 1 5 6 5 4 1 3 3 4 4 2 3 2	NO

This example is represented by the diagram on the right, which, no matter what tunnels Ali digs, the cats can force him to lose.



G. Magic and Merch

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

Parsa has been practising magic tricks for days — he wants to impress Krystalle enough that she'll agree to design new MAPS merch for him. His favourite trick involves a line of cards laid out on the table.

Whenever Parsa waves his hand over a card, the card flips: if it was face-up, it becomes face-down; if it was face-down, it becomes face-up.

Parsa plans to perform for t nights in a row. Each night, a fresh set of cards is placed in front of him. With one move, he can choose any consecutive block of cards and flip them all at once.

To make his performance truly dazzling (and hopefully win Krystalle's agreement), Parsa wants to finish each night with all the cards face-up. Now he wonders: what is the minimum number of moves he must perform each night to achieve this?

Input

The first line of input contains one integer:

- t ($1 \le t \le 1000$) – the number of days Parsa will perform his magic tricks.

Each of the following t lines contains a string s_i. In this string, the j-th character is 1 if the j-th card is face-up, and 0 if the j-th card is face-down.

- $|s_i|$ ($0 \le |s_i| \le 50$) – number of cards in day i

Output

Print a single integer, the minimum number of moves Parsa must perform.

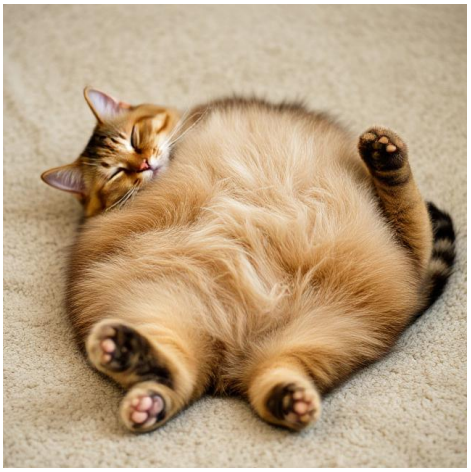
Examples

standard input	standard output
5 01 00010 11011 111000 101010101	1 2 1 1 4

This page is intentionally left blank

H. Greedy Cat

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

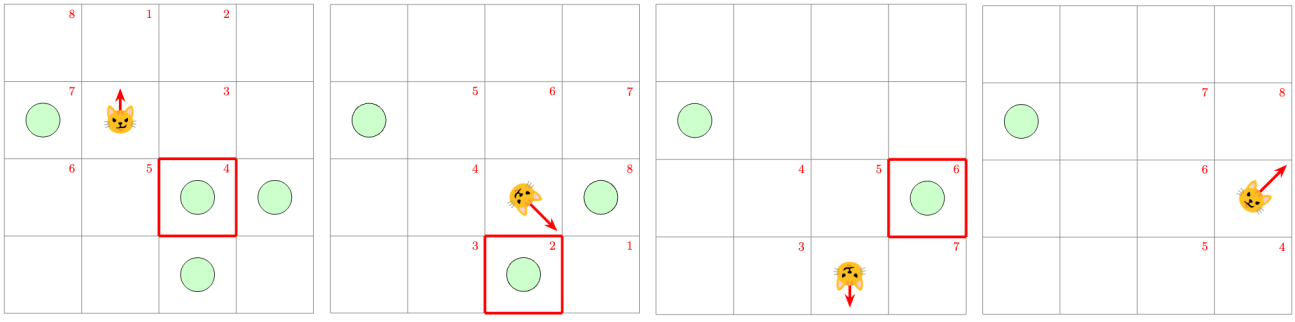


After his battle through the tunnels, Ali takes one of the cats (the smirking one) as his pet, so it now resides in Ali's house, which due to his love of competitive programming he's had architected as an $n \times m$ grid.

This cat is a particularly dangerous combination of greedy and lazy. They love treats, and will jump at the opportunity to eat one... but only if it is in close vicinity.

In particular:

0. The cat will begin facing upwards.
1. The cat will scan all 8 adjacent grid positions from its facing direction clockwise, until it finds a treat. (See the images below for an example of the 8 positions.)
2. It will rotate to face the treat, and move to this grid square.
3. Go back to step 1, scanning for an adjacent treat.
4. If no treats are found, then the cat goes to sleep (Game Over).



Pictured: The cat's movement through example test 1.

Ali would like to know, given the current position of the cat and the positions of the treats, where they will end up, and how many treats they will eat.

Input

Input will begin with two integers n ($1 \leq n \leq 100$) and m ($1 \leq m \leq 100$), the height and width of the grid. n lines will follow, detailing the grid. Each line will contain m characters, based on the following legend:

- `C` is the position of the cat (this will only occur once in the input)
- `T` represents a treat
- `.` represents an empty position

The cat will always face upwards to begin with.

Output

Output 3 space-separated integers:

1. How many treats the cat will eat before going to sleep
2. The column of the cat's final position (this is 1 on the leftmost column, and m on the rightmost)
3. The row of the cat's final position (this is 1 on the bottommost row, and n on the topmost)

Examples

standard input	standard output
4 4 .... TC.. ..TT ..T.	3 4 2
4 6 ..TTT. .T.TT. .TC... T.TTT.	8 1 1

I. GiDi-Up

Input file:	`standard input`
Output file:	`standard output`
Time limit:	4 seconds
Memory limit:	512 megabytes

In order to fund his battles, Ali starts a chariot-share service called GiDi-Up.

Since this is occurring in medieval times, village-to-village rides don't take hours, but days to months. And so intermediary tavern stays are a necessity.

Each village-to-village connection is bidirectional and takes a full day to traverse, and he'd like to get the patron from village 1 to village n in the least days possible.

Ali's particular service is all expenses paid, so anytime he crosses through a village with a tavern, he is obligated to let the patron stay there free of charge, adding 1 full day to his journey (and the patron *always* wants a free tavern stay). He is therefore incentivised to find his way through villages that don't have taverns.

That being said, he doesn't want to get a bad review from a noble and end up on the wrong end of a guillotine. If the patron spends more than k days on the road without seeing a tavern, then it'll be off with his head.

Can you help figure out how to get Ali and his patron from 1 to n efficiently, without getting a bad review?

Input

The first line will contain four integers:

- n ($2 \leq n \leq 10^5$), the number of villages,
- t ($1 \leq t \leq n$), the number of villages that contain a tavern,
- r ($1 \leq r \leq 10^5$), the number of village to village connections that exist,
- k ($1 \leq k \leq 5$), the number of days the patron can go without sleeping at a tavern.

The next line will contain t integers t_i ($1 \leq t_i \leq n$), detailing which villages have a tavern.

The next r lines will contain 2 integers, representing the start and end villages for each bidirectional connection.

Output

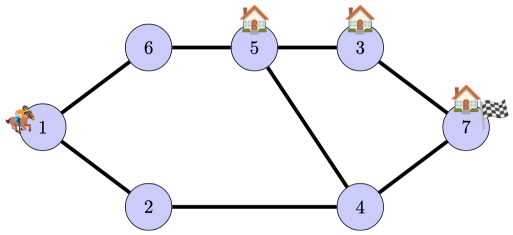
Your output should consist of only a single integer. If the trip is impossible within the constraints given, output -1.

Otherwise, print the minimum number of days required for the journey.

Examples

standard input	standard output
7 3 8 2 3 5 6 1 6 6 5 1 2 4 2 4 5 5 3 7 3 4 7	6

This is represented by the following diagram:

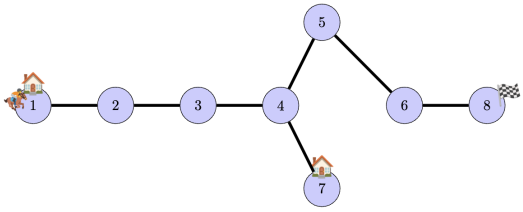


The optimal path takes 6 days to complete.

We can do this by traversing from village 1 to village 6 to village 5, where we have to stay at the tavern, then to village 4, and then to village 7 (where we then also need to stay for the night at the tavern).

standard input	standard output
8 2 7 5 1 7 2 1 2 3 3 4 7 4 4 5 6 5 8 6	10

This is represented by the following diagram:



The optimal path takes 10 days to complete.

First we have to stay at the tavern in the first village, then we can begin by traversing from village 1 to village 2 to village 3 to village 4 to village 7, where we have to stay at the tavern, then to village 4, 5, 6 and 8.

J. Graduation

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 second
Memory limit:	256 megabytes

Lauren has just graduated, and to celebrate, she promised to throw a series of parties for her friends. For each party, she knows exactly how much food she needs: on the i-th night, she must provide at least p_i slices of pizza and b_i burger sliders.

The local diner offers only three kinds of combo meals, each with its own price:

- A Pizza Combo, which contains two slices of pizza.
- A Burger Combo, which contains two burger sliders.
- A Mixed Combo, which contains one slice of pizza and one burger slider.

Lauren wants to celebrate her graduation in style without overspending. Your task is to help her calculate the minimum total cost needed for each party night so that all food requirements are met.

Lauren can buy any number of any combo meal, as long as she has enough food for the party.

Input

The first line of input contains one integer:

- t ($1 \leq t \leq 100{,}000$) – the number of graduation parties Lauren will host.

For each party i:

- The first line contains two integers p_i and b_i ($0 \leq p_i, b_i \leq 1{,}000{,}000$), the minimum number of pizza slices and burger sliders required for that night.
- The second line contains three integers P, B, and M ($1 \leq P, B, M \leq 1000$), the prices of a Pizza Combo, Burger Combo, and Mixed Combo, respectively.

Output

For each party, print a single integer — the minimum cost Lauren must pay to satisfy her friends.

Examples

standard input	standard output
4 2 2 1 2 2 1 5 10 1 2 3 1 7 7 3 0 1 5 3 1	3 4 9 1

K. Birthday

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	256 megabytes

It's Winson's birthday! Since he's the president of MAPS, Phoebe and Michael have decided to surprise him with a giant cake. But before they can decorate it, they stumble upon a fun little challenge.

They have a pack of $4 \times n$ matchsticks. They want to use all the matchsticks to build several disjoint, non-nested squares.

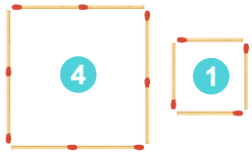
A square with side length s uses $4 \times s$ matchsticks and has area s^2.

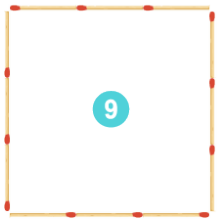
Equivalently, if the squares have side lengths $s_1, s_2, \ldots, s_k$ (positive integers), then

- $s_1 + s_2 + \ldots + s_k = n$
- total area $= {s_1}^2 + {s_2}^2 + \ldots + {s_k}^2$

Now Michael wants a configuration with the **smallest possible total area** but Phoebe wants a configuration with the **largest possible total area**.

For example, if $n = 3$ they can build the squares like:





They ask you: Given n, what are the smallest and largest possible sums of the areas of the squares?

Input

The first and only line of input contains one integer:

- n $(1 \leq n \leq 100)$ – meaning they have $4 \times n$ matchsticks.

Output

Print two lines, each with one integer: the first line being the smallest possible sum of the areas of the squares, and the second line being the largest possible sum of the areas of the squares.

Examples

standard input	standard output
3	3 9

standard input	standard output
1	1 1

standard input	standard output
4	4 16