TCS CODEVITA ZONE 1 ALL QUESTIONS

1. Diet Plan

Problem Description

Arnold is planning to follow a diet suggested by his Nutritionist. The Nutritionist prescribed him the total protein, carbohydrates and fats, he should take daily. Arnold searches on an online food store and makes a list of protein, carbohydrates and fats contained in a single unit of various food items.

His target is to have the maximum protein, carbohydrates and fats in his diet without exceeding the prescribed limit. He also wants to have as much diverse food items as much as possible. That is, he does not want to have many units of one food item and 0 of others. Multiple combinations of 'units of food items' are possible to achieve the target. Mathematically speaking, diversity is more if the difference between the number of units of food item chosen the most and the number of units of another food item chosen the least, is as small as possible.

To solve this problem, he uses maximum possible number of units of all the items so that total amount of protein, carbohydrates and fats is not more than prescribed limit. For example - if total nutrition required is 100P, 130C and 130F (where P is Protein, C is Carbohydrates and F is Fats) and 2 different food items, viz. Item A and Item B, have following amount of nutrition:

Item A - 10P, 20C, 30F

Item B - 20P, 30C, 20F

then, he can have (maximum possible) 2 units of all the items as having 3 units will exceed the prescribed amount of Carbohydrates and fats.

Next, he chooses food items to fulfill the remaining nutrients. He chooses one more units of maximum number of food items. He continues this process till he cannot add a unit of any food item to his diet without exceeding the prescribed limit. In this example, he can choose one more unit of item B or one more unit of item A. In case he has two sets of food items then the priority is given to fulfill the requirements of Protein, Carbohydrates, and Fats in that order. So he chooses item B.

You will be provided the maximum nutrients required and the nutrients available in various food items. You need to find the amount of nutrients for which there is a shortfall as compared to the prescription, after making his selection using the process described above. In the example he still needs 20P, 0C, 10F to achieve his target.

Constraints

Number of Food Items <= 10

Maximum amount of Nutrients is less than 1500 i.e. x +y + z <= 1500

Amount of P, C, F in two food items will not be same

P, C, F values in input can be in any order

Output should be in order - P, C, F.

Input

First line contains the maximum limit of nutrients in the following format.

xP yC zF, where x, y and z are integers

Second line contains nutrient composition of different food items separated by pipe (|).

Output

Print the shortfall for each nutrient type, fulfilled separated by space.

E.g. If the output is 10P, 20 C, 30 F, then print "10 20 30" (without quotes).

Time Limit

1

Examples

Example 1

Input

100P 130C 130F

10P 20C 30F|20P 30C 20F

Output

20 0 10

Explanation

Having 2 units of item A and 3 units of item B provides - 2 * [10P 20C 30F] + 3 * [20P 30C 20F] = 100P, 130C, 120F. This is the best combination that can reduce the shortfall [20P, 0C, 10F] without exceeding his prescription. In contrast, if 3 units of A and 2 units of B are chosen then 3 * [10P 20C 30F] + 2 * [20P 30C 20F] = 70P, 120C, 130F produces a shortfall of [30P, 10C, 0F]. However, since protein shortfall in this case is more than the previous combination, this is not the right combination to follow.

Example 2

Input

130P 120C 110F

4P 9C 2F|4P 3C 2F|7P 1C 3F

Output

2 4 50

Explanation

Having 9 units of item A, 9 units of item B and 8 units of Item C provides - 9 * [4P 9C 2F] + 9 * [4P 3C 2F] + 8 * [7P 1C 3F] = 108P, 116C, 60F. This is the best combination that can reduce the shortfall [2P, 4C, 50F] without exceeding his prescription.

---

2. Signal Connection

Problem Description

You have been given longitude and latitude of locations from where the channels are going to be broadcasted. You also get the height of tower from which these channels are broadcasted. Moreover, you have been given the location of your friend Jason. You have to calculate how many connections he can make to the towers. Take Radius of earth= 6371 KM.

All the computation has to be accurate up to 6 digits after the decimal point.

Constraints

1 <= N < 10^5

Input

First line contains integer N denoting the number of locations from where the channel is going to be broadcasted

Second line contains N space-separated decimal values denoting latitudes

Third line contains N space-separated decimal values denoting longitudes

Fourth line contains N space-separated integer values denoting the height of tower from which channels are broadcasted

Fifth Line contains two space-separated decimal values denoting latitude, longitude of Jason's location

Output

Print the number of channels Jason can connect with

Time Limit

1

Examples

Example 1

Input

2

19.076090 17.387140

72.877426 78.491684

2 1

18.516726 73.856255

Output

1

Explanation

First latitude and longitude is Mumbai and second is for Hyderabad from where the channel signals are broadcasted. Jason is in Pune. According to signal strength Jason will only be able to connect Mumbai tower.

Example 2

Input

2

28.644800 22.572645

77.216721 88.363892

5 7

48.864716 2.349014

Output

0

Explanation

First latitude and longitude is Delhi and second is for Kolkata from where the channel signals are broadcasted. Jason is in Paris. According to signal strength Jason will not be able to connect any tower.

3. Faulty Keyboard

Problem Description

Mr. Wick has a faulty keyboard. Some of the keys of the keyboard don't work. So he has copied all those characters corresponding to the faulty keys on a clipboard. Whenever those characters need to be typed he pastes it from the clipboard. In typing whatever is required he needs to make use of paste, backspace and cursor traversal operations. Help him in minimize the number of operations he needs to do to complete his typing assignment. Each operation has one unit weightage.

Constraints

1 <= S <= 16

1 <= T<= 10^4

String T and S will only be comprised of letters a-z and digits 0-9

Input

First line contains text T to be typed Second line contains string S of all the faulty keys pasted on clipboard

Output

Print the minimum number of operations required for typing the text T

Time Limit

1

Examples

Example 1

Input

experience was ultimate

ew

Output

14

Explanation

experience =(2+2+2+2) =[{p+b} + {p+b} +{p+b} +{p+b}]

was=(4)=[p+m+b+m]

ultimate=(2)=[p+b]

where p=paste, b=backspace ,m= move cursor

Example 2

Input

supreme court is the highest judicial court

su

Output

17

Explanation

supreme =(1) =[p]

court=(4)=[p+m+b+m]

is=(2)=[p+b]

the=(0)

highest=(2)=[p+b]

judicial=(4)=[p+m+b+m]

court=(4)=[p+m+b+m]

4. Engagement Ring

Problem Description

Sejal was on a month-long vacation to Europe and has a return trip to India from Lisbon, a city in south west part of Europe. However, a day before the return flight she realizes that she lost her engagement ring. After much contemplation, she decides to go to all the cities she visited to find her ring.

She maps all the cities she visited on a graph with Lisbon being at point (0,0). She then makes a route plan to visit all the cities and return to Lisbon by taking the shortest possible distance. She does not remember having her ring even in the first city she visited so there are high chances that she may have lost her ring in the initial part of her trip also. In case there are more than one routes which have the shortest possible distance, she picks that route in which the first city she visited, comes first. For example, if she visited cities 2,5,7,1,8,3 in that order and routes 0,1,8,3,2,5,7,0 and 0,8,3,1,5,2,7,0 (0 being Lisbon) have the same shortest possible distance then she will choose route 0,1,8,3,2,5,7,0 because she visited city 1 before city 8.

Her travel guide, Harry also offers to help Sejal. Sejal asks him to travel separately on the same route, but in reverse direction such that each city is visited only once. They plan to travel 20 Kms in each city on taxi to search the ring. Inter-city travel is done on trains only.

A secret service officer knows the coordinates of the city that Sejal visited during her the trip in that order. He also knows the city in which the ring is lost but will inform Sejal or Harry only when one of the two is in that city.

He knows the path that Sejal has drawn to visit all the cities and return to Lisbon. With Sejal and Harry following that path and either one of them reaching the city where the ring is lost, the secret service officer will inform the person in that city, that the ring will be 10 km away from their current location. They will travel back 10 km in the same city, to catch the train back to Lisbon. Calculate the total distance traveled by Sejal in her search and her return to Lisbon from that city.

If the ring is found by Sejal, she goes back to Lisbon from that city. If the ring is found by Harry, he informs Sejal on call at that point. If Sejal is in a city (searching in taxi) she returns to Lisbon via train from that city (without searching any further in that city). If Sejal is on train, she will need to complete the journey and then return from that other city. If the call comes at the exact point she is taking a train, she can return from that city itself.

Each unit in the graph is equal to 1 Km. Assume the speed of all trains and taxis is same. Do not consider the decimal values while calculating the distance between two cities, ie. distance will be the floor of the calculated distance.

Constraints

Floor of the value is to be used while calculating distance between cities

Each city is connected to every other city via trains

Total number of cities <10

Input

First Line will provide the coordinates (x|y) of cities separated by semicolon (;)

Second Line will provide the number of city where the ring is found

Output

One integer representing the number total distance traveled by Sejal

Time Limit

1

Examples

Example 1

Input

0|90;90|90;90|0

2

Output

347

Explanation

Since routes 1,2,3 and 3,2,1 will give the shortest distance, she will select route 1,2,3 as she visited city 1 before city 3. However, Harry will follow the opposite path - 3,2,1. They both will be in city 2 when they will find it. Total distance Sejal covers will be Lisbon to city 1 + 20Kms, City 1 to City 2 + 10 km +10 km and then City 2 to Lisbon. i.e. 90 + 20 + 90 + 20 +127 = 347 km

Example 2

Input

10|70;30|30;80|20;120|75;90|120

3

Output

334

Explanation

Sejal will choose the route 1, 5, 4, 3, 2 with minimum distance of 378 km. However, Harry will follow the opposite path - and will find the ring in City 3. The moment Harry finds the ring, Sejal will be travelling from City 1 to City 5. So she will complete the journey till City 5 and return from there to Lisbon. So the total distance covered by Sejal will be 70+20+94+150 = 334 km

---

5. Maximum Prize

Problem Description

Imagine you are a martial arts fighter fighting with fellow martial artists to win prize money. However unlike traditional competitions, here you have the opportunity to pick and choose your opponent to maximize your prize kitty. The rules of maximization of prize kitty are as follows

► You have a superpower bestowed upon you, that you will win against anyone you challenge

► You have to choose the right order because unfortunately the superpower does not ensure that your prize money is always the highest

► Every victory against an opponent that you challenge and win against, will translate into a certain winning sum

► Here begins the technical part that you need to know in order to maximize your winning prize money

- All your opponents are standing in one line next to each other i.e. the order of opponents is fixed
- Your first task is to choose a suitable opponent from this line
- When you choose one opponent from that line, he steps out of the line and fights you.
- After you beat him, you get to decide how your prize money for winning against him will be calculated
- Essentially, if the opponent you have beaten has two neighbours, then you have the option to multiply the *opponent number* with any one of the two neighbours and add the other *opponent number*. That value becomes your prize money for that match
- If your opponent has only one neighbor then your prize money for that match is product of current *opponent number* with neighbours' *opponent number*
- When dealing with last opponent in the tournament, your prize money is equal to the value of the last *opponent number*
- As the tournament proceeds, the opponent that you have beaten has to leave the tournament

Example: 2 5 6 7

This depicts that you have four opponents with numbers 2 5 6 and 7 respectively

1. Suppose you choose to fight opponent number 5, then after winning, the max prize kitty you can win for that match is = 5*6+2 = 32

   Now opponent number 5 is out of the game. So opponent number 2 6 7 remain

2. Suppose you now choose to fight opponent number 2, then after winning, the max prize kitty you can win for that match is = 2*6+0 = 12. Your overall prize kitty is now 32 + 12 = 44

   Now opponent number 2 is out of the game. So opponent number 6 7 remain

3. Suppose you now choose to fight opponent number 6, then after winning, the max prize kitty you can win for that match is = 7*6+0 = 42. Your overall prize kitty is now 44 + 42 = 86

   Now opponent number 6 is out of the game. So opponent number 7 remains

4. After beating opponent number 7, the max prize kitty you can win for that match is 7

   So overall prize kitty in this case is 93.

Other orders of choosing opponents will yield the following overall prize kitty

- Order 7->2->6->5 will yield overall prize kitty as 87
- Order 2->5->6->7 will yield overall prize kitty as 88
- Order 5->6->2->7 will yield overall prize kitty as 95
- Order 6- >7->2->5 will yield overall prize kitty as 97
- But by following the order 6->5->2->7 will yield overall prize kitty as 105, which is maximum.

Your task is to maximize your prize kitty by taking the right decisions

Input

First line contains an integer N which denotes the number of opponents in the tournament

Second line contains N space separated integers, which are the *opponent numbers* of other opponents

Output

Print the maximum number of coins you can win

Constraints

1 <= N <= 500

0 <= individual coin count < 100

Time Limit

1

Examples

Example 1

Input

4

2 5 6 7

Output

105

Explanation:

Refer the explanation in problem description.

Example 2

Input

3

7 8 9

Output

151

Explanation:

1. You choose to fight opponent number 8, then after winning, the max prize kitty you can win for that match is = 8*9+7 = 79

Now opponent number 8 is out of the game. So opponent number 7 9 remain

2. Suppose you now choose to fight opponent number 7, then after winning, the max prize kitty you can win for that match is = 7*9+0 = 63. Your overall prize kitty is now 79 + 63 = 142

Now opponent number 7 is out of the game. So opponent number 9 remains

3. After beating opponent number 9, the max prize kitty you can win for that match is 9

So overall prize kitty in this case is 142 + 9 = 151.

---

6. Constellation

Problem Description

Three characters { #, *, . } represents a constellation of stars and galaxies in space. Each galaxy is demarcated by # characters. There can be one or many stars in a given galaxy. Stars can only be in shape of vowels { A, E, I, O, U } . A collection of * in the shape of the vowels is a star. A star is contained in a 3x3 block. Stars cannot be overlapping. The dot(.) character denotes empty space.

Given 3xN matrix comprising of { #, *, . } character, find the galaxy and stars within them.

Note: Please pay attention to how vowel ***A*** is denoted in a 3x3 block in the examples section below.

Constraints

3 <= N <= 10^5

Input

Input consists of single integer N denoting number of columns.

Output

Output contains vowels (stars) in order of their occurrence within the given galaxy. Galaxy itself is represented by # character.

Time Limit

1

Examples

Example 1

Input

18

* . * # * * * # * * * # * * * . * .

* . * # * . * # . * . # * * * * * *

* * * # * * * # * * * # * * * * . *

Output

U#O#I#EA

Explanation

As it can be seen that the stars make the image of the alphabets U, O, I, E and A respectively.

Example 2

Input

12

* . * # . * * * # . * .

* . * # . . * . # * * *

* * * # . * * * # * . *

Output

U#I#A

Explanation

As it can be seen that the stars make the image of the alphabet U, I and A.

7. Number Distancing

Problem Description

Consider 9 natural numbers arranged in a 3x3 matrix:

n11 n12 n13

n21 n22 n23

n31 n32 n33

Define numbers "in contact" with a given number to be those that appear closest to it on the same row, column or diagonally across:

Contacts of number n11: n12, n22 and n21

Contacts of number n12: n11, n21, n22, n23, n13

Contacts of number n13: n12, n22, n23

Contacts of number n21: n11, n12, n22, n32, n31

Contacts of number n22: n11, n12, n13, n23, n33, n32, n31, n21

Contacts of number n23: n13, n12, n22, n32, n33

Contacts of number n31: n21, n22, n32

Contacts of number n32: n31, n21, n22, n23, n33

Contacts of number n33: n32, n22, n23

The problem now is that numbers having a common factor (other than 1) should not be "in contact". In other words, a pair of numbers can remain neighbours only if their highest common factor is 1.

The following rules apply to enforce this "distancing":

1. The central number (n22) stays put.

2. The corner numbers (n11, n13, n33, n31) can move in the same row or column or diagonally away from the centre.

3. The numbers "on the walls" (n12, n23, n32, n21) can only move from the walls i.e. n21 can only move "left", n12 can only move "up", n23 can only move "right" and n32 can only move "down".

4. Each number should stay put as far as possible and the "distancing" operation should result in the least number of numbers ending up without any contacts.

5. After satisfying rule 4, if there are multiple options for the final matrix, then the "distancing" operation should result in the smallest (m x n matrix, including the intervening blank space elements, with the least possible value of m*n).

6. If, after satisfying all the rules above, there are multiple distancing options for a set of numbers, the largest number keeps to its original cell.

Constraints

1 <= Element of grid <= 100

Input

First line consists of 9 space separated integers denoting n11, n12, n13, ... n23, n33 respectively.

Output

Print the "contact" less numbers in ascending order of their value separated by space. Output "None" if there are no such numbers.

Time Limit

1

Examples

Example 1

Input

23 33 12 1 2 5 25 6 10

Output

10

Explanation

Initial configuration

23 33 12

1 2 5

25 6 10

The optimal distancing options result in the following possibility (space denoted by *):

23 33 * 12

1 2 5 *

25 * * *

* 6 * 10

10 ends up as the number without contacts.

Example 2

Input

1 2 3 4 5 6 7 8 9

Output

None

Explanation

Initial configuration:

1 2 3

4 5 6

7 8 9

The optimal distancing options result in the following 5x3 matrix (space denoted by *):

* 2 3

1 * *

4 5 6

7 * *

* 8 9

There is finally no number without a contact.

Example 3

Input

2 6 2 10 19 12 2 20 2

Output

2 2 2 2 10 12

Explanation

Initial Configuration:

Approach 1) Moving 6 and 20

Final Matrix

2 * 6 * 2

* * * * *

* 10 19 12 *

* * * * *

2 * 20 * 2

Approach 2) Moving 10 and 12

2 * * * 2

* * 6 * *

10 * 19 * 12

* * 20 * *

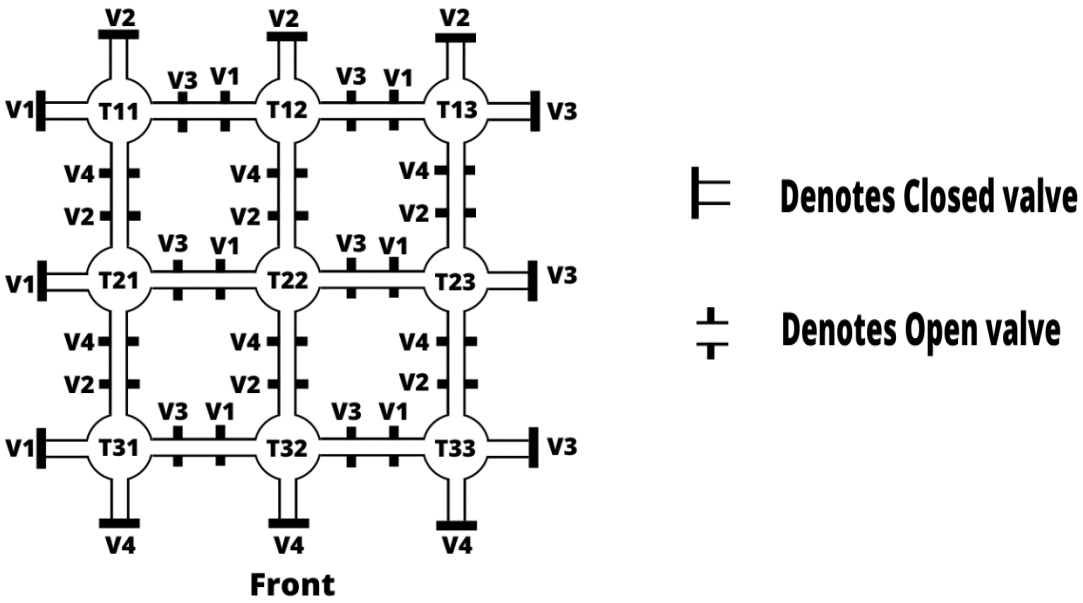
2 * * * 2

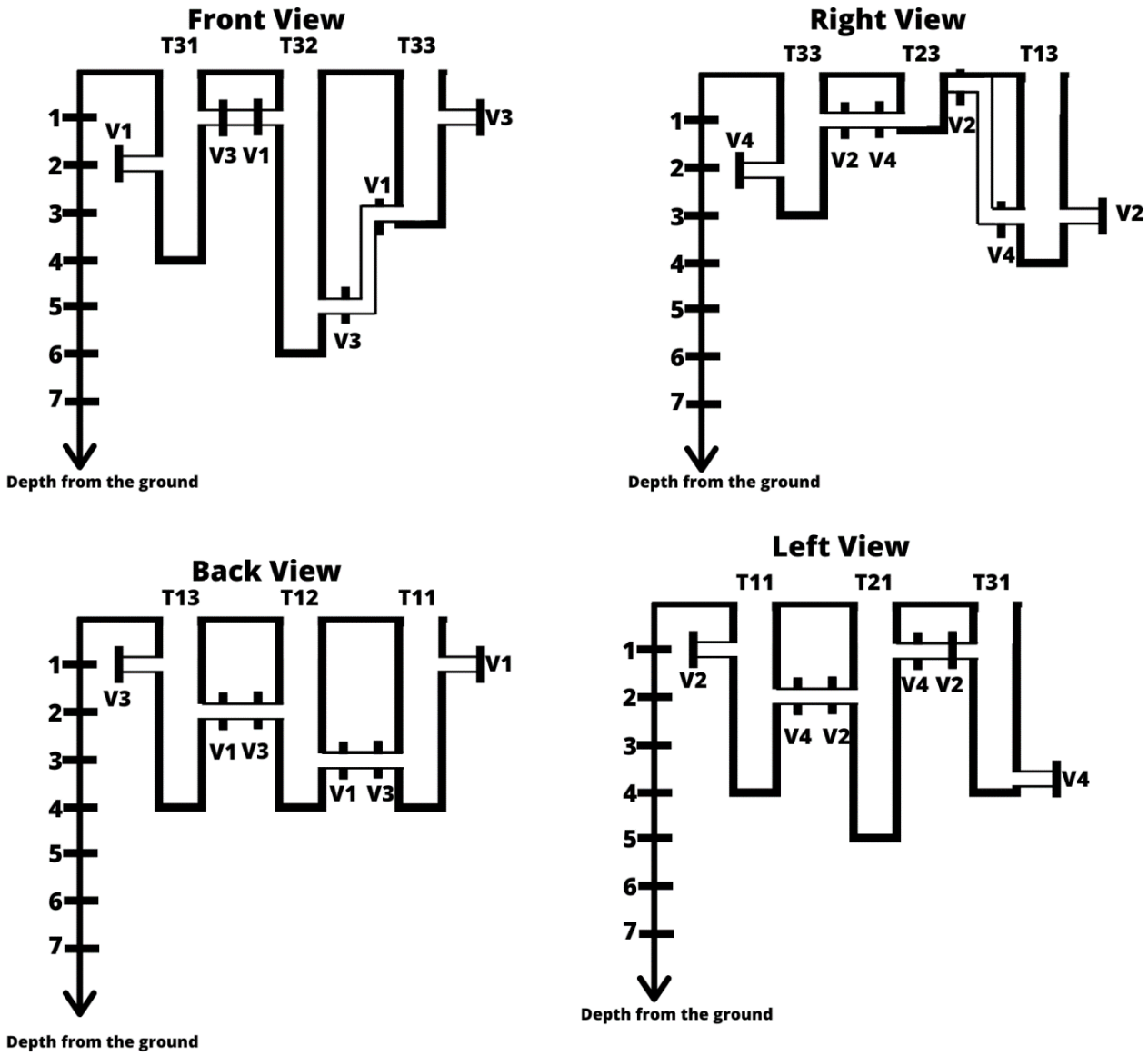
We prefer approach 2) and not 1) since the largest of all the elements (20) needs to be retained in it's original cell.

8. Tank Network

Problem Description

There are M*N underground water tanks, which are arranged in M rows and N columns. Tanks are labelled as Tij, where i is the row number and j is the column number (1 <= i <= M and 1 <= j <= N). Each tank is of certain depth beginning from the ground level (measured in feet). Each tank has four valves situated at a certain depth from ground level. All 4 valves can be at different depth from the ground level. Valves are orthogonal to each other. Valves of each tank are labelled as V1, V2, V3 and V4. Valves may be open or closed. If a valve is closed, no water flows out through it. If a valve is open, water flows out through it. Each valve is connected to the valve of nearby tank using connection tubes (or it is terminated if there is no tank nearby). All tanks are of same shape. For all tanks, 1 liter water occupy 1 foot height of the tank. For example, below diagram depicts a 3*3 tank network with all tanks empty.





When water starts filling into the tank, the valves attached to different tanks will determine the flow of water within the tank network.

You need to find the amount of water (in liters) required to fill a given tank (say, T_{xy} - x^{th} row and y^{th} column) to a particular height h, given details about tank dimensions. You can ignore the amount of water getting filled in valves and connection tubes.

Constraints

1 <= m <= 350

1 <= n <= 350

1 <= depth of tank <= 100

0 <= position of valve <= 100

Input

First line contains 2 space separated integers denoting M and N.

Next M lines, each contains N space separated integers forming an M*N matrix denoting the depth (in feet) of each tank.

First integer of first line in this M*N block denotes the depth of $Tank_{11}$, second integer denotes the depth of $Tank_{12...}$
First integer of second line in this M*N block denotes the depth of $Tank_{21}$, second integer denotes the depth of $Tank_{22...}$

.

.

First integer of M^{th} line in this M*N block denotes the depth of $Tank_{M1}$, second integer denotes the depth of $Tank_{M2...}$

Next M lines, each contains N-space separated integers forming an M*N matrix denoting the depth (in feet) at which valve V1 of each tank is situated.

First integer of first line in this M*N block denotes the depth of valve V1 for $Tank_{11}$, second integer denotes the depth of valve V1 for $Tank_{12...}$
First integer of second line in this M*N block denotes the depth of valve V1 for $Tank_{21}$, second integer denotes the depth of valve V1 for $Tank_{22...}$

.

.

First integer of M^{th} line in this M*N block denotes the depth of valve V1 for $Tank_{M1}$, second integer denotes the depth of valve V1 for $Tank_{M2...}$

Next M lines, each contains N space separated integers forming an M*N matrix denoting the depth (in feet) at which valve V2 of each tank is situated.

First integer of first line in this M*N block denotes the depth of valve V2 for $Tank_{11}$, second integer denotes the depth of valve V2 for $Tank_{12...}$
First integer of second line in this M*N block denotes the depth of valve V2 for $Tank_{21}$, second integer denotes the depth of valve V2 for $Tank_{22...}$

.

.

First integer of M^{th} line in this M*N block denotes the depth of valve V2 for $Tank_{M1}$, second integer denotes the depth of valve V2 for $Tank_{M2...}$

Next M lines, each contains N space separated integers forming an M*N matrix denoting the depth (in feet) at which valve V3 of each tank is situated.

First integer of first line in this M*N block denotes the depth of valve V3 for $Tank_{11}$, second integer denotes the depth of valve V3 for $Tank_{12...}$
First integer of second line in this M*N block denotes the depth of valve V3 for $Tank_{21}$, second integer denotes the depth of valve V3 for $Tank_{22...}$

.

.

First integer of M^{th} line in this M*N block denotes the depth of valve V3 for $Tank_{M1}$, second integer denotes the depth of valve V3 for $Tank_{M2...}$

Next M lines, each contains N-space separated integers forming an M*N matrix denoting the depth (in feet) at which valve V4 of each tank is situated.

First integer of first line in this M*N block denotes the depth of valve V4 for $Tank_{11}$, second integer denotes the depth of valve V4 for $Tank_{12...}$
First integer of second line in this M*N block denotes the depth of valve V4 for $Tank_{21}$, second integer denotes the depth of valve V4 for $Tank_{22...}$

.

.

First integer of M^{th} line in this M*N block denotes the depth of valve V4 for $Tank_{M1}$, second integer denotes the depth of valve V4 for $Tank_{M2...}$

Next M lines, each contain N space separated integers forming an M*N matrix denoting whether the status of valve V1 i.e. { open (0), closed (1) }.

First integer of first line in this M*N block denotes the status of valve V1 for $Tank_{11}$, second integer denotes the status of valve V1 for $Tank_{12...}$
First integer of second line in this M*N block denotes the status of valve V1 for $Tank_{21}$, second integer denotes the status of valve V1 for $Tank_{22...}$

.

.

First integer of M^{th} line in this M*N block denotes the status of valve V1 for $Tank_{M1}$, second integer denotes the status of valve V1 for $Tank_{M2...}$

Next M lines, each contain N space separated integers forming an M*N matrix denoting whether the status of valve V2 i.e. { open (0), closed (1) }.

First integer of first line in this M*N block denotes the status of valve V2 for $Tank_{11}$, second integer denotes the status of valve V2 for $Tank_{12...}$
First integer of second line in this M*N block denotes the status of valve V2 for $Tank_{21}$, second integer denotes the status of valve V2 for $Tank_{22...}$

.

.

First integer of M^{th} line in this M*N block denotes the status of valve V2 for $Tank_{M1}$, second integer denotes the status of valve V2 for $Tank_{M2...}$

Next M lines, each contain N space separated integers forming an M*N matrix denoting whether the status of valve V3 i.e. { open (0), closed (1) }.

First integer of first line in this M*N block denotes the status of valve V3 for $Tank_{11}$, second integer denotes the status of valve V3 for $Tank_{12...}$
First integer of second line in this M*N block denotes the status of valve V3 for $Tank_{21}$, second integer denotes the status of valve V3 for $Tank_{22...}$

.

.

First integer of M^{th} line in this M*N block denotes the status of valve V3 for $Tank_{M1}$, second integer denotes the status of valve V3 for $Tank_{M2}$...

Next M lines, each contain N space separated integers forming an M*N matrix denoting whether the status of valve V1 i.e. { open (0), closed (1) }.

First integer of first line in this M*N block denotes the status of valve V4 for $Tank_{11}$, second integer denotes the status of valve V4 for $Tank_{12}$...
First integer of second line in this M*N block denotes the status of valve V4 for $Tank_{21}$, second integer denotes the status of valve V4 for $Tank_{22}$...

.

.

First integer of M^{th} line in this M*N block denotes the status of valve V4 for $Tank_{M1}$, second integer denotes the status of valve V4 for $Tank_{M2}$...
Next line contain space separated 3 integers denoting x, y and h i.e. tank T_{xy} filled to depth h

Output

Single line containing an integer denoting the amount of water in liters.

Time Limit

1

Examples

Example 1

Input

2 2

3 4

4 3

1 2

2 3

1 3

2 1

3 1

1 1

2 3

4 2

1 0

1 0

1 1

1 0

0 1

0 1

0 0

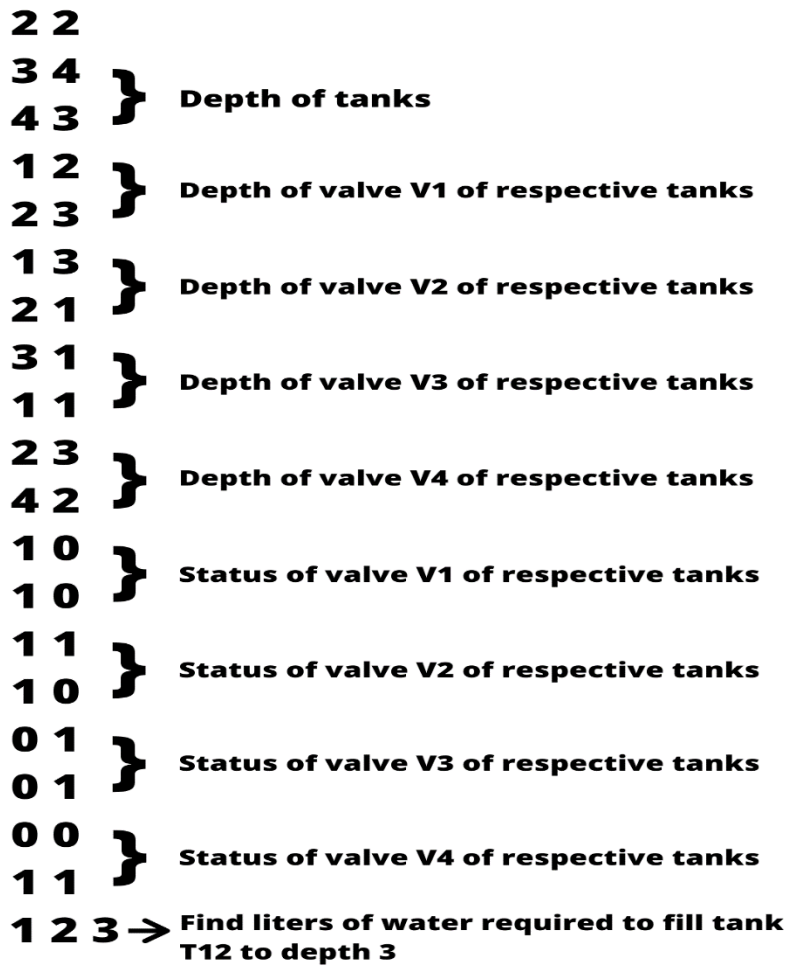
1 1

1 2 3

Output

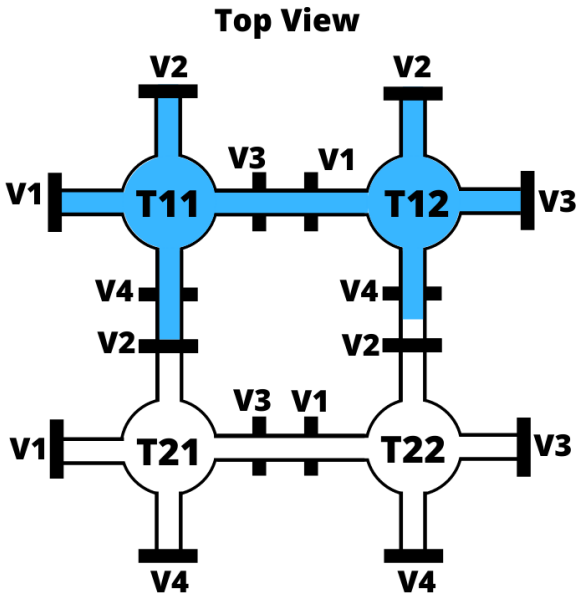
5

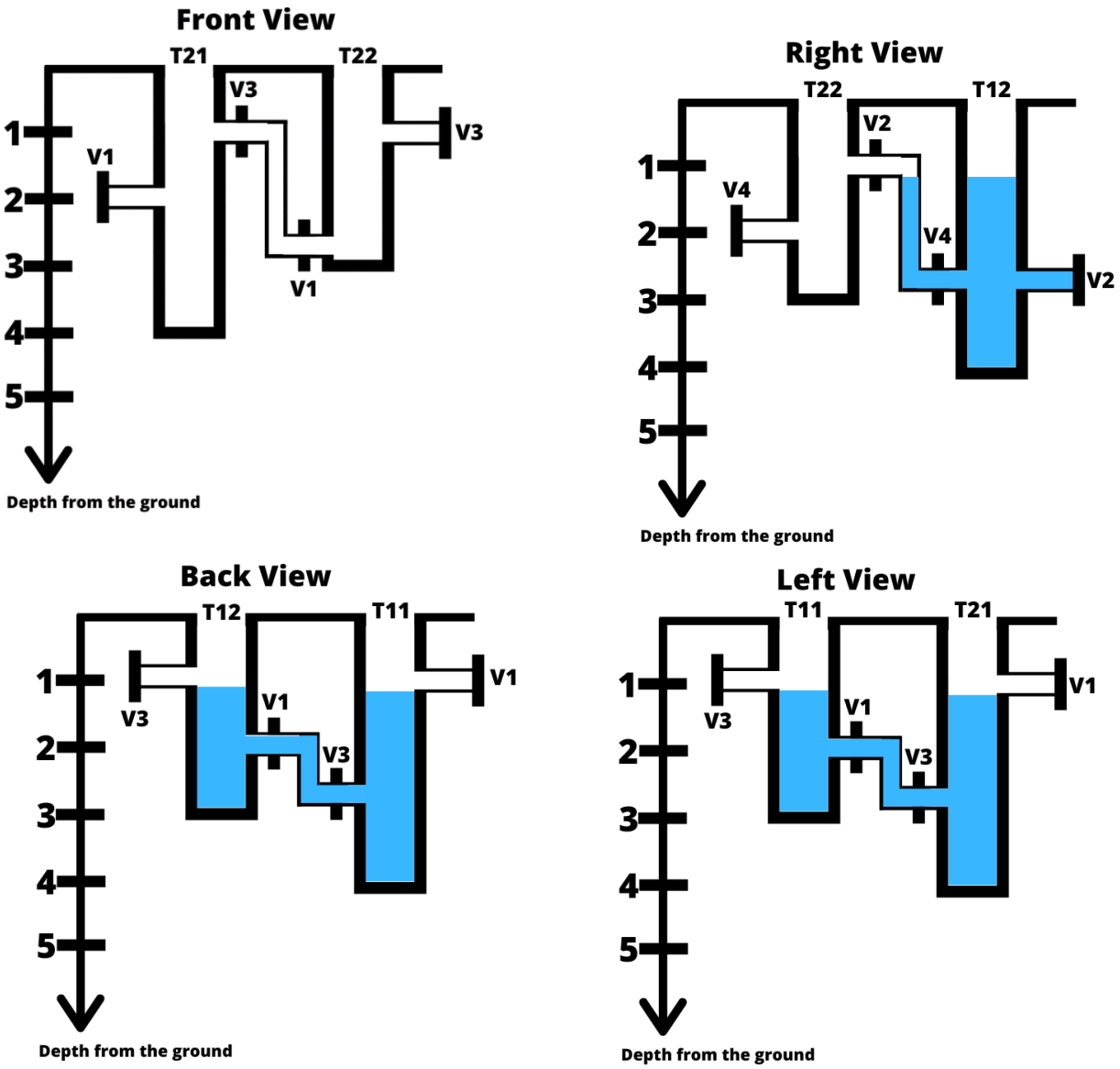
Explanation :



From first input line it's clear that there are four tanks - T_{11}, T_{12}, T_{21}, Tank $_{22}$, arranged in 2 rows and 2 columns.
From input lines 2 and 3, its clear that the depths of T_{11} is 3 feet, T_{12} is 4 feet, T_{21} is 4 feet and Tank $_{22}$ is 3 feet.

From input lines 4 and 5, its clear that valve V1 of T_{11} is located at a depth of 1 foot, that of T_{12} is at 2 feet, that of T_{21} is at 2 feet, and that of Tank $_{22}$ is at 3 foot.
From input lines 6 and 7, its clear that valve V2 of T_{11} is located at a depth of 1 foot, that of T_{12} is at 3 feet, that of T_{21} is at 2 feet, and that of Tank $_{22}$ is at 1 foot.
From input lines 8 and 9, its clear that valve V3 of T_{11} is located at a depth of 3 feet, that of T_{12} is at 1 feet, that of T_{21} is at 1 foot, and that of Tank $_{22}$ is at 1 foot.
From input lines 10 and 11, its clear that valve V4 of T_{11} is located at a depth of 2 feet, that of T_{12} is at 3 feet, that of T_{21} is at 4 foot, and that of Tank $_{22}$ is at 2 foot.
From input lines 12 and 13, its clear that the valve V1 of T_{11} is closed, that of T_{12} is opened, that of T_{21} is closed, and that of Tank $_{22}$ is opened.
From input lines 14 and 15, its clear that the valve V2 of T_{11} is closed, that of T_{12} is closed, that of T_{21} is closed, and that of Tank $_{22}$ is opened.
From input lines 16 and 17, its clear that the valve V3 of T_{11} is opened, that of T_{12} is closed, that of T_{21} is opened, and that of Tank $_{22}$ is closed.
From input lines 18 and 19, its clear that the valve V4 of T_{11} is opened, that of T_{12} is opened, that of T_{21} is closed, and that of Tank $_{22}$ is closed.
From input line 20, its clear that we need to find the amount of water required to fill the tank T_{12} for a height of 3 feet.
After filling the tank T_{12} to 3 feet, the tank network might look like below figure (shaded colour denotes water).





From the above diagram, it's clear that tanks T_{12} and T_{11} got filled with 3 and 2 liters of water respectively. So, total 5 liters of water is required to fill the tank T_{12} to a height of 3 feet.

Example 2

Input

3 3

6 6 6

6 6 6

6 6 6

1 1 1

1 1 1

1 1 1

1 1 1

1 1 1

1 1 1

1 1 1

1 1 1

1 1 1

1 1 1

1 1 1

1 1 1

1 0 0

1 0 0

1 0 0

1 1 1

0 0 0

0 0 0

0 0 1

0 0 1

0 0 1

0 0 0

0 0 0

1 1 1

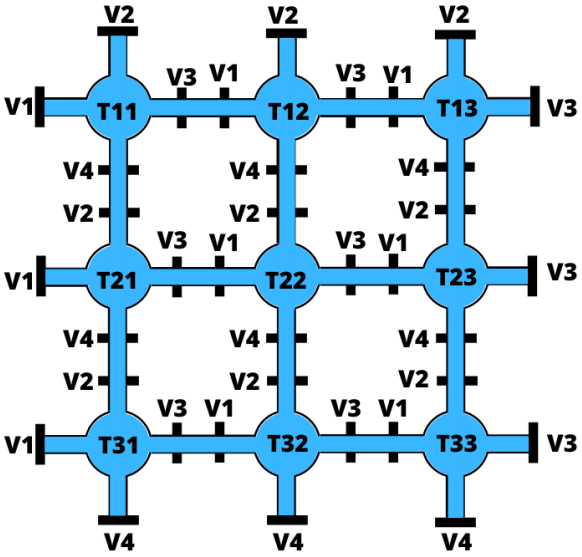
1 2 6

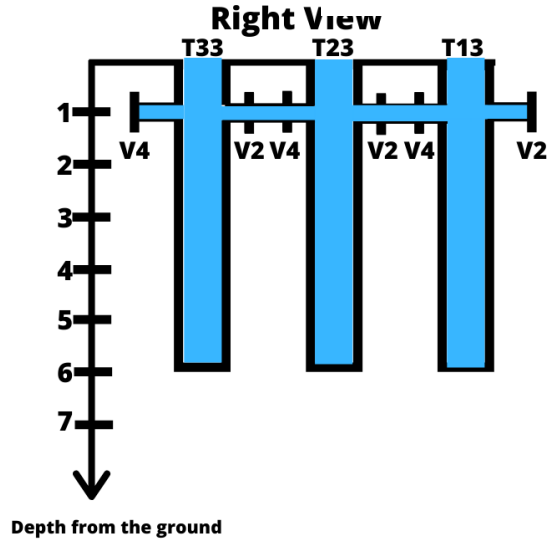
Output

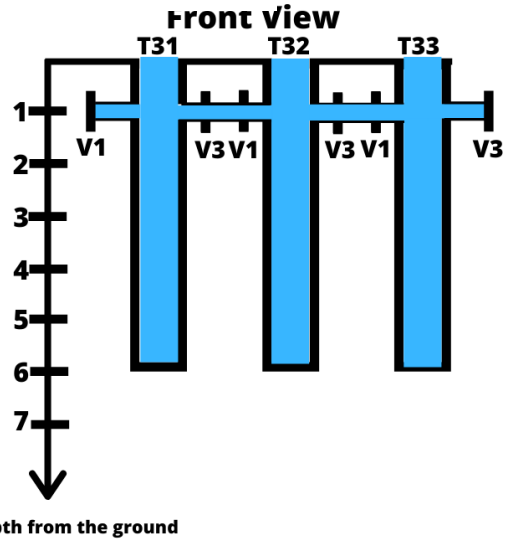
54

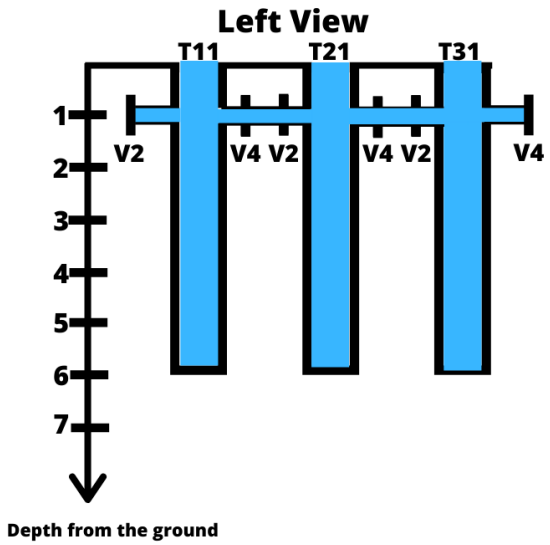
Explanation :

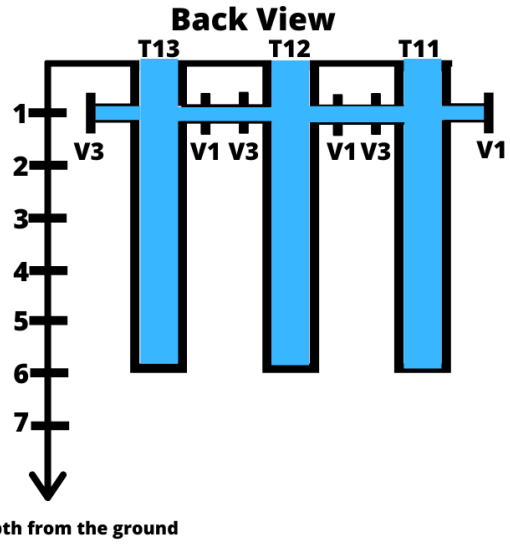
After filling the tank T_{12} to 6 feet, the tank network might look like below figure











From the above diagram, it's clear that all tanks got filled with 6 liters of water. So, total 54 liters of water is required to fill the tank T_{12} to a height of 6 feet.

9. Critical Planets

Problem Description

War between Republic and Separatist is escalating. The Separatist are on a new offensive. They have started blocking the path between the republic planets (represented by integers), so that these planets surrender due to the shortage of food and supplies. The Jedi council has taken a note of the situation and they have assigned Jedi Knight Skywalker and his Padawan Ahsoka to save the critical planets from blockade (Those planets or system of planets which can be accessed by only one path and may be lost if that path is blocked by separatist).

Skywalker is preparing with the clone army to defend the critical paths. He has assigned Ahsoka to find the critical planets. Help Ahsoka to find the critical planets(C) in ascending order. You only need to specify those planets which have only one path between them and they cannot be accessed by any other alternative path if the only path is compromised.

Constraints

M <= 10000

N <= 7000

Input

First line contains two space separated integers M and N, where M denotes the number of paths between planets and N denotes the number of planets.

Next M lines, each contains two space separated integers, representing the planet numbers that have a path between them.

Output

C lines containing one integer representing the critical planet that they need to save in ascending order of the planet number if no planet is critical then print -1

Time Limit

1

Examples

Example 1

Input

3 4

0 1

1 2

2 3

Output

0

1

2

3

Explanation

Since all the planets are connected with one path and cannot be accessed by any alternative paths hence all the planets are critical.

Example 2

Input

7 6

0 2

0 1

1 2

2 3

4 5

3 4

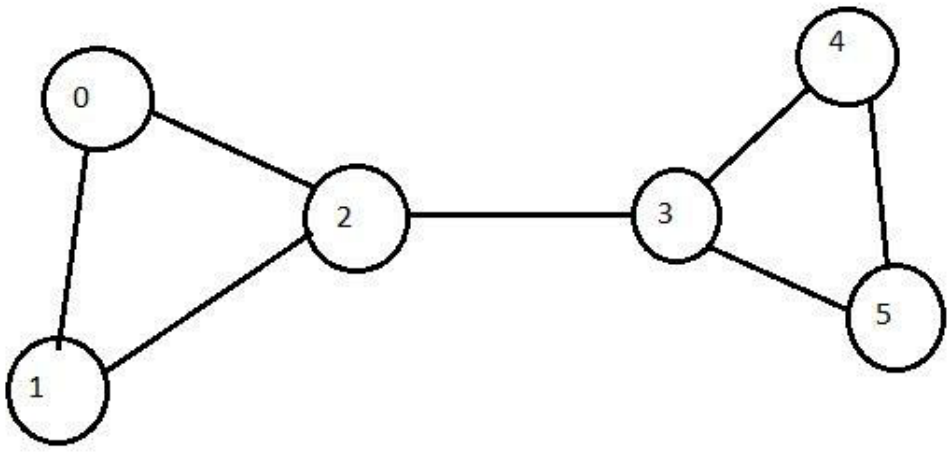
3 5

Output

2

3

Explanation



If the republic loose the path between 2 and 3 then the two system of planets will not be able to communicate with each other. Hence 2 and 3 are critical planets.

10. Lift

Problem Description

In a building there are some lifts. Optimize the allocation of lifts.

Say there are N requests and M lifts. Minimize the maximum request waiting time.

Rules of lift allocation

1) One needs to assign indexes to lifts. i.e. decide lift #1, lift #2, lift #3, etc apriori.

2) Since all N requests are known apriori decide the initial (at time t = 0) location of lift such that it will help in minimizing waiting time.

3) Even if all the N requests time are known apriori, other than initial time, i.e. t = 0, the waiting lifts cannot be moved towards target floor until the request is actually issued.

4) After a request is issued for optimality calculation, even a lift is in transit it can be considered for serving that request.

5) If at any moment, more than one lift can serve the request with same waiting time, give preference to the lift with lower index. i.e. If Lift #2 and Lift #4 can serve a particular request in 2 seconds, then prefer Lift #2 over Lift #4.

6) Neglect the time required to get in and get out of a lift.

7) Once a lift starts serving a request it would not stop in between. If would finally stop at destination of that request.

8) The speed of lift is 1 floor/second.

Constraints

0 <= T <= 86400.

0 <= S, D <= 100.

1 <= N, M <= 1000.

Input

First line contains 2 integers, N and M, representing the number of requests and number of lifts in the building respectively.

Next N lines contain 3 integers, T, S, and D, representing the timestamp of request, source floor, destination floor respectively.

Output

Print single integer representing the waiting time

Time Limit

1

Examples

Example 1

Input

3 2

0 2 3

4 2 5

6 7 3

Output

3

Explanation

There are 3 requests and 2 lifts.

Initially at time t = 0, both the lifts, lift #1 and lift #2 will be at floor 2 (Initial allocation).

Request #1 and Request #2 can be responded instantly, i.e. waiting time = 0.

Lift #1 will get unallocated at floor 3 at time t = 1.

Lift #1 can move only after the next request is actually received at time t = 4, if required. Relate this with Rule #3 mentioned in problem description.

Lift #2 will get unallocated at floor 5 at time t = (4 + 3) = 7.

Now, if Lift #1 is used to respond request #3, waiting time would be 4 seconds since Lift#1 is at floor #3 and will need 4 seconds to reach floor #7.

In this case, waiting time of all requests - {0,0,4} and the maximum waiting time is 4.

Instead, if Lift #2 is used to respond request #3, waiting time would be calculated as follows

Lift #2 will get unallocated at time t = 7, so we will have to wait 7-6 = 1 seconds for lift #2 to complete it's previous request.

Time needed for Lift #2 to travel from floor #5 to floor #7 = 7-5 = 2 seconds. Therefore, total waiting time = 1 + 2 = 3 seconds.

In this case, waiting time of all requests - {0, 0, 3} and the maximum waiting time is 3.

As we have to minimize the maximum waiting time, since 2nd option is yielding lesser waiting time than 1st option, 2nd option is the answer.

Therefore, the output is 3.

Example 2

Input

5 3

0 2 3

4 2 5

6 7 3

3 5 6

2 5 7

Output

1

Explanation

There are 5 requests and 3 lifts.

Initially, at t = 0, lift #1 will be at floor #2 whereas lift #2 and #3 will be at floor#5 (Initial allocation).

Request #1, #4, #5 can be responded instantly, i.e. waiting time = 0.

Lift #1 will get unallocated at floor 3 at time t = 1.

Lift #2 will get unallocated at floor 7 at time t = 2 + 2 = 4.

Lift #3 will get unallocated at floor 6 at time t = 3 + 1 = 4.

Request #2 can be served by lift #1. Waiting time would be 2 - 1 = 1.

Now, lift #1 will get unallocated at floor #5 at time t = 4 + 1 + 3 = 8.

Request #3 can be served by lift #3 as it will be floor #6 at t = 4. Waiting time = 0.

Waiting time of all requests - {0, 1, 0, 0, 0}.

Therefore, the output is 1.

11. SudoKube

Problem Description

John, a research scholar / Professor / Puzzle solver wants your help in publishing his work on SudoKube on his online blog for his followers and students.

A SudoKube is a mixture of Rubics cube and Sudoku. A SudoKube has exactly 6 appearances of every digit from 1 to 9 across the cube, whereas Rubics cube has 6 different colours.

As John wants to publish his work in text /document form (no video) he's concerned how he would depict the step by step work of rotation in 2D form. Following are the notions and concepts John follows:

1. The six faces of the cube are named FRONT, BACK, UP, DOWN, LEFT and RIGHT respectively.

2. Just like a Rubics cube which move in 90 and 180 degrees in both clockwise and anti clockwise directions, so can the SudoKube

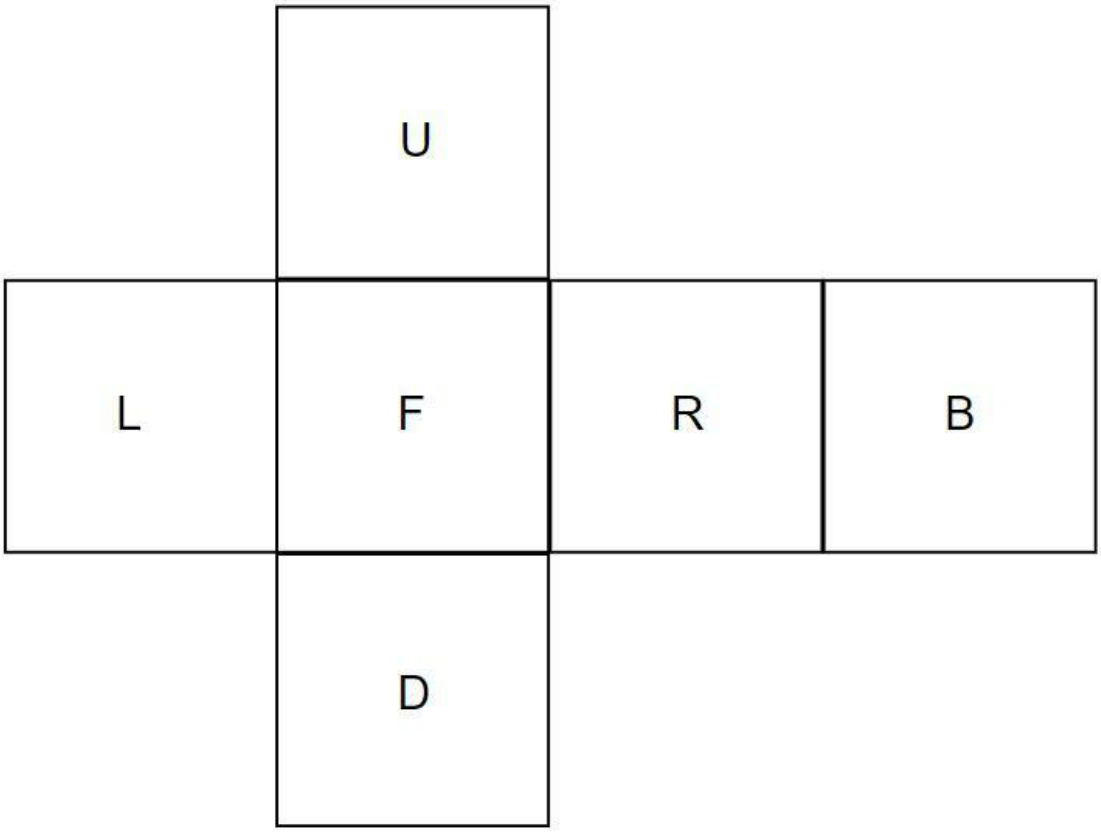
3. Any given face of the cube is a 3x3 square matrix whose indices are denoted by (0,0) to (2,2). Diagram below illustrates the same.

4. An elementary move is denoted in the following fashion.

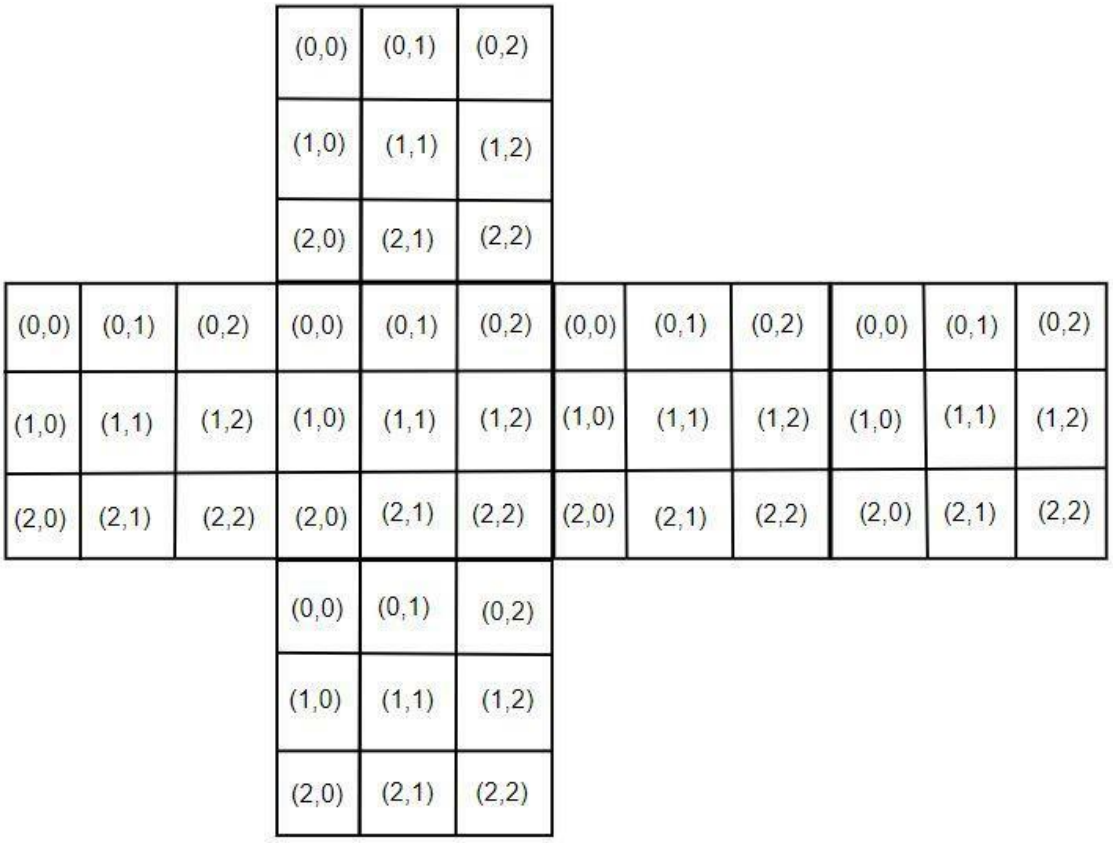
   i. If a given face is rotated by 90 degrees clockwise about the axis passing from the centre of the face to the centre of the cube, the move is denoted by the first letter of the name of the face.

   ii. If the rotation is anticlockwise by 90 degrees, the letter is followed by an apostrophe (').

   iii. If the rotation is by 180 degrees, the letter is followed by a 2.



Above image display the position of the faces



Above diagram displays the indices of the matrix on the faces

John wants to test his notations on you. He has given you the initial position of the SudoKube and he has given you a set of operations to be performed on the SudoKube basis his notation. After applying all the operations, the final SudoKube state should be the same as what John expects. Your task is to apply the operations and print the final SudoKube state.

Constraints

Values in SudoKube will be between 1 and 9

No of moves < 15

Input

• First eighteen lines contain the values of the faces on SudoKube in the order given below

D D D

D D D

D D D

U U U

U U U

U U U

L L L

L L L

L L L

F F F

F F F

F F F

R R R

R R R

R R R

B B B

B B B

B B B

where

- D for Down face
- U for upper face
- L for Left face
- F for Front face
- R for Right face
- B for Back face.

Input contains digits from 1 to 9 instead of letters; letters are displayed for better understanding of the faces and the expected input format

• Nineteenth line contains a sequence of space delimited moves that need to be performed on the SudoKube
Example 1: D F2 R' U - to understand this please refer second example from the *Examples* section below

Example 2: L2 U B F' D2 R - lets understand how to interpret this set of operations

- L2 means rotate the Left side by 180 degrees
- U means rotate the Up side by 90 degrees clockwise
- B means rotate the Back side by 90 degrees clockwise
- F' means rotate the Front side by 90 degrees anticlockwise
- D2 means rotate the Down side by 180 degrees
- R means rotate the Right side by 90 degrees clockwise

In summary, first eighteen lines denotes the state of the SudoKube, 19th line denotes the operation to be performed on that state and output should be the resulting state.

Output

Print 3x3 matrix corresponding to the order (D, U, L, F, R, B). Between every 3x3 matrix there should be a new line.

Time Limit

1

Examples

Example 1

Input

4 7 1

2 8 7

6 3 5

5 8 3

3 1 6

9 4 2

5 2 4

3 7 8

5 1 9

6 1 4

9 4 8

2 5 7

7 9 1

1 9 6

6 2 8

8 6 3

7 2 5

3 9 4

F

Output

6 1 7

2 8 7

6 3 5

5 8 3

3 1 6

9 8 4

5 2 4

3 7 7

5 1 1

2 9 6

5 4 1

7 8 4

9 9 1

4 9 6

2 2 8

8 6 3

7 2 5

3 9 4

Explanation:

The output shows the state of SudoKube when the front side is rotated clockwise

Example 2

Input

4 7 1

2 8 7

6 3 5

5 8 3

3 1 6

9 4 2

5 2 4

3 7 8

5 1 9

6 1 4

9 4 8

2 5 7

7 9 1

1 9 6

6 2 8

8 6 3

7 2 5

3 9 4

D F2 R' U

Output

2 4 5

3 8 9

5 7 6

4 3 5

2 1 8

8 7 6

9 1 3

3 7 1

3 9 7

1 6 7

8 4 6

4 1 6

1 6 3

9 9 5

4 8 4

5 2 2

7 2 5

9 2 8

Explanation

The above output prints the state of cube after D F2 R' U operation are performed. Here

· D means rotate the Down side by 90 degrees clockwise

· F2 means rotate the Front side by 180 degrees

· R' means rotate the Right side by 90 degrees anti clockwise

· U means rotate the Up side by 90 degrees clockwise

12. Codekart

Problem Description

Codu wants to create a shopping application. The application would sell only SHIRT and SHOE and have a cost that can be modified based on market needs. This application should allow users in two roles, viz. store manager(SM) and shopper(S).

Codu wants to test the app. He wants the application to execute a few commands and print the output.

Following is the list of allowed **commands**:

CMD SM ADD [ITEM NAME] [ITEM QTY] - adds the given quantity of the item to the inventory and prints(returns) the quantity added, otherwise prints -1 when there is any error or invalid input. Item qty can only be whole numbers > 0.

CMD SM REMOVE [ITEM NAME] - removes the item from the inventory, returns and prints -1 when there is an error, otherwise prints(returns) 1.

CMD SM GET_QTY [ITEM NAME] - returns and prints the currently available quantity for the item in the inventory, otherwise prints(returns) 0 in case the item is not found.

CMD SM INCR [ITEM NAME] [ITEM QTY] - adds the given quantity of the item to the inventory and prints(returns) the quantity added, otherwise prints(returns) -1 when there is any error or invalid input. item qty can only be whole numbers > 0.

CMD SM DCR [ITEM NAME] [ITEM QTY] - removes the given quantity of the item from the inventory and prints(returns) the quantity added, otherwise prints(returns) -1 when there is any error or invalid input. item qty can only be whole numbers > 0.

CMD SM SET_COST [ITEM NAME] [COST] - sets the cost of the item, returns the value, otherwise prints -1 in case of any errors or invalid input. Cost must be decimal.

CMD S ADD [ITEM NAME] [ITEM QTY] - adds the given quantity of the item to the shopping cart and prints(returns) the quantity added, otherwise prints -1 when there is any error or invalid input. Item qty can only be whole numbers > 0.

CMD S REMOVE [ITEM NAME] - removes the item from the shopping cart, returns and prints -1 when there is an error, otherwise prints(returns) 1.

CMD S INCR [ITEM NAME] [ITEM QTY] - adds the given quantity of the item to the shopping cart and prints(returns) the quantity added, otherwise prints(returns) -1 when there is any error or invalid input. item qty can only be whole numbers > 0.

CMD S DCR [ITEM NAME] [ITEM QTY] - removes the given quantity of the item from the shopping cart and prints(returns) the quantity added, otherwise prints(returns) -1 when there is any error or invalid input. item qty can only be whole numbers > 0.

CMD S GET_ORDER_AMOUNT - gets the total price of the items in the cart, returns the value, otherwise prints -1 in case of any error or invalid input. The total amount should be rounded and printed up to two decimal places.

NOTE- Increment and Decrement operations are only possible when the item is already in the inventory or cart. If increment or decrement is attempted on items that do not exist in the cart, then the command should return and print -1.

If an attempt is made to add an item that is already in the inventory or cart, such operations should result in an error and must return and print -1.

If an item which is present in the cart of inventory is removed using *remove* command and an increment or decrement operation is performed on it, such operations should result in an error and must return and print -1.

If any item quantity after decrement becomes zero, the same is removed from the corresponding inventory or cart. Performing increment or decrement operation after such a previous decrement operation, should result in an error and return -1.

You need to think of other similar error conditions while implementing the solution.

Please note at the beginning of a test case or command set, both the inventory as well as the cart is empty.

Here,

SM= STORE MANAGER

S= SHOPPER

You are required to create the application for Codu to manage the shopping kiosk.

The first line of input **T**, gives the number of test cases.

Each test set is a set of commands, which ends with "END" string.

Each command in a test case is on a new line

Constraints

1<=T<=10

Input

First line contains an integer T, which denotes the number of test cases

Second line onwards, there will be commands until we receive END command. Any command after the END command belongs to next test case.

For command format refer to Example section.

Output

Print output of every command. (Print double value for command SET_COST(rounding to one decimal places) and GET_ORDER_AMOUNT(rounding to two decimal places))

Time Limit

1

Examples

Example 1

Input

1

CMD SM SET_COST SHOE 5

CMD SM SET_COST SHIRT 10

CMD SM ADD SHOE 5

CMD SM ADD SHIRT 10

CMD SM DCR SHIRT 5

CMD SM INCR SHOE 5

CMD SM GET_QTY SHIRT

CMD SM GET_QTY SHOE

CMD SM REMOVE SHIRT

CMD SM GET_QTY SHIRT

CMD S ADD SHOE 2

CMD S INCR SHOE 2

CMD S DCR SHOE 1

CMD S GET_ORDER_AMOUNT

END

Output

5.0

10.0

5

10

5

5

5

10

1

0

2

2

1

15.00

Explanation :

From commands "**CMD SM SET_COST SHOE 5**" and "**CMD SM SET_COST SHIRT 10**"

We are successfully setting the cost as 5.0 and 10.0 respectively.

From next commands "**CMD SM ADD SHOE 5**" and "**CMD SM ADD SHIRT 10"**

Quantity of 5 shoes and 10 shirts has been successfully added to the inventory.

From next commands "**CMD SM DCR SHIRT 5**"" and "**CMD SM INCR SHOE 5"**

Shirt quantity is decremented by 5 and shoe quantity is incremented by 5. This leaves us with 5 shirts and 10 shoes in the inventory.

From next commands "**CMD SM GET_QTY SHIRT**" and "**CMD SM GET_QTY SHOE"**

We are getting the quantity of shirt and shoe, which is 5 and 10 respectively.

From next command "**CMD SM REMOVE SHIRT"**

Shirt is removed from the inventory and hence 1 is printed.

From next command "**CMD SM GET_QTY SHIRT"**

We are querying the quantity of shirt, which is 0 as it was removed in the previous command.

From next command "**CMD S ADD SHOE 2**"

Shopper adds two shoes to the cart hence 2 is printed.

From next commands "**CMD S INCR SHOE 2**" and "**CMD S DCR SHOE 1**"

The user increments these shoes by 2 and then decrements by 1 hence 2 and 1 are printed. SO current shoes in cart= 2+2-1=3

From next commands "**CMD S GET_ORDER_AMOUNT**"

The next command asks to print order amount or cart value, which is the cost of shoes * the number of shoes=5*3=15 hence 15.00 is printed by rounding to two decimal places.

13. Paste Reduction

Problem Description

Some keys in Codu's computer keyboard are not working. Fortunately for Codu, these characters are present in a previously existing text file.

He wants to write a paragraph which involves typing those characters whose keys are defunct in Codu's keyboard. Only option left for Codu is to copy-paste those characters from the previously existing text files. However, copy-pasting keys is a laborious operation since one has to switch windows and also previously copied items are lost once a new set of characters are copied. Hence, Codu wants to minimize the number of times he needs to copy-paste from that text file. Fortunately there can be situations where previously copied characters are readily available for pasting. Help Codu devise a method to minimize the number of times a paste operation is needed, given - the text he intends to type and the faulty keys

Constraints

0 < Length of paragraph <= 1000 characters

0 < number of faulty keys in keyboard <= 36

Only a to z and 0 to 9 keys can be faulty.

Input paragraph will not contain any upper-case letter.

Input

First line contains a paragraph that is be to be written.

Second line contains a string. This string has to be interpreted in the following fashion

All characters in that string correspond to faulty keys

That string is available for copy as-is from the other text file that Codu is referring

Also, individual characters can always be copied from the same text file

Output

Single integer denoting minimum number of copy operations required

Time Limit

1

Examples

Example 1

Input

supreme court is the highest judicial court

su

Output

4

Explanation

Codu will first paste su from the file when typing characters su in the word supreme.

In the second instance, Codu will need to copy character u and paste it when typing character u in the word court.

In the third instance, Codu will need to copy character su and paste su when typing character s in the word is. Codu will back track using the left arrow key and type the characters "the highe".

In the fourth instance, Codu will again paste su. The overall string typed until this moment is "supreme court is the highesuu". Codu will then back track again using left arrow key and type characters "t j". At this point the typed string is "supreme court is the highest juu". Cursor is after character j. Codu will use right arrow key and now the cursor will be after ju. Codu will now type "udicial co". String typed till this point is "supreme court is the highest judicial cou". Cursor is after character o. Codu will use right arrow and the cursor will be after character u. Finally Codu will type "rt".

Final string - "supreme court is the highest judicial court" - is thus fully typed. Here, the number of paste operations are 4.

14. Closing Value

Problem Description

Stock Exchange of country XYZ is still working on pen paper mode, wherein the traders have to bid the price for buying and selling the stocks and the stocks prices are checked manually, then the buying and selling data is validated and if the condition matches then it is recorded in the record book. For example, if A wants to buy stocks at 100 and B is willing to sell the required stocks at 95, then A can buy his desired share at 95 and the price of the share will become 95. The price of shares of a company is determined by the latest transaction recorded in the record book.

As the number of transactions are increasing it is getting hard to match and record the transactions manually.

The stock exchange wants to go online and has hired Karim to make the process online. The stock exchange personnel will give him the bid and he has to design a program to match the values and if the transaction is done, then record it in the record book which will also be online.

Help Karim in recording the completed transactions and enter the closing price in record book. He has to give the closing values of all the companies whose transactions have been recorded in ascending order.

In case, if more than one matching bids are received then the trader with lower TraderId will get the preference. In case only a partial match is received then it will be a split transaction. For example, consider following transaction below:

TraderId	TradeType	StockName	Price	Quantity
1	Buy	ABC	50	90
2	Buy	ABC	50	90
3	Sell	ABC	50	100

Trader 3 will sell his 90 stocks to Trader 1 at price of 50. Since Trader 3 has 10 stocks left, the remaining stocks can be purchased by Trader 2. Hence, Trader 1 and Trader 3 have squared off their positions whereas as Trader 2 has open bid for 80 stocks of ABC at a price of 5.

Here, Trader 3 had a split transaction which is closed whereas Trader 2 also had split transaction which is still open.

Constraints

0 < N <=10000

Input

First line contains an integer N, denoting the number of bids

Next N lines, each contains 5 space separated values denoting <TraderId, Tradetype, StockName, Price, Quantity>

Where,

TraderId is single integer containing the id of trader, smaller trader id indicates the timestamp of the trade

Tradetype can be either Buy/Sell

StockName is the name of the company that trader wants to bid

Price indicates the price at which the trader has bid

Quantity indicates the number of stocks trader wants to purchase/buy

Output

Lexicographically ascending order of Stock Name(S) along with respective values(C) in the format (S:C). If no company has traded the stock, then print "Stocks not traded"

Time Limit

1

Examples

Example 1

Input

3

1 Sell ABC 1876 173

2 Sell DEF 7160 221

3 Buy ABC 6986 864

Output

ABC:1876

Explanation

Here transaction of ABC is recorded but the DEF is not recorded as no one has bid to buy the stocks of DEF.

Trader with id 3 will buy 173 shares from trader with id 1 of Stock ABC at unit price of 1876. Since this becomes the closing price of ABC, output is ABC:1876

Example 2

Input

3

1 Sell ABC 1876 173

2 Sell DEF 7160 221

3 Buy ABC 1200 864

Output

Stocks not traded

Explanation

Since, no transaction has taken place as the buy bid is less than sell for company ABC and there is no trade to buy DEF, there is no record in the record book. Hence "Stocks not traded" will be the output.

15. Prime Time Again

Problem Description

Here on earth, our 24-hour day is composed of two parts, each of 12 hours. Each hour in each part has a corresponding hour in the other part separated by 12 hours: the hour essentially measures the duration since the start of the day part. For example, 1 hour in the first part of the day is equivalent to 13, which is 1 hour into the second part of the day.

Now, consider the equivalent hours that are both prime numbers. We have 3 such instances for a 24-hour 2-part day:

5~17

7~19

11~23

Accept two natural numbers D, P >1 corresponding respectively to number of hours per day and number of parts in a day separated by a space. D should be divisible by P, meaning that the number of hours per part (D/P) should be a natural number. Calculate the number of instances of equivalent prime hours. Output zero if there is no such instance. Note that we require each equivalent hour in each part in a day to be a prime number.

Example:

Input: 24 2

Output: 3 (We have 3 instances of equivalent prime hours: 5~17, 7~19 and 11~23.)

Constraints

10 <= D < 500

2 <= P < 50

Input

Single line consists of two space separated integers, D and P corresponding to number of hours per day and number of parts in a day respectively

Output

Output must be a single number, corresponding to the number of instances of equivalent prime number, as described above

Time Limit

1

Examples

Example 1

Input

36 3

Output

2

Explanation

In the given test case D = 36 and P = 3

Duration of each day part = 12

2~14~X

3~15~X

5~17~29 - instance of equivalent prime hours

7~19~31 - instance of equivalent prime hours

11~23~X

Hence the answers is 2.

Example 2

Input

49 7

Output

0

Explanation

Duration of each day part = 7

2~9~X~23~X~37~X

3~X~17~X~31~X~X

5~X~19~X~X~X~47

7~X~X~X~X~X~X

Hence there are no equivalent prime hours.

16. Minimize The Sum

Problem Description

Given an array of integers, perform atmost K operations so that the sum of elements of final array is minimum. An operation is defined as follows -

Consider any 1 element from the array, arr[i].

Replace arr[i] by floor(arr[i]/2).

Perform next operations on updated array.

The task is to minimize the sum after atmost K operations.

Constraints

1 <= N, K <= 10^5.

Input

First line contains two integers N and K representing size of array and maximum numbers of operations that can be performed on the array respectively.

Second line contains N space separated integers denoting the elements of the array, arr.

Output

Print a single integer denoting the minimum sum of the final array.

Time Limit

1

Examples

Example 1

Input

4 3

20 7 5 4

Output

17

Explanation

Operation 1 -> Select 20. Replace it by 10.

New array = [10, 7, 5, 4]

Operation 2 -> Select 10. Replace it by 5.

New array = [5, 7, 5, 4].

Operation 3 -> Select 7. Replace it by 3.

New array = [5,3,5,4].

Sum = 17.

17. Travel Cost

Problem Description

Suresh wants to travel from city A to city B. But due to coronavirus pandemic, almost all the cities have levied entry tax into the cities so that the number of people entering the city can be limited. Suresh can skip at the most m cities at a time. Suresh has to declare his itinerary at the time of leaving city A. Thus he will have to pay upfront for entire itinerary and also has to pay a fee to get the slips issued. Upon payment he will be given the slips for intermediate cities where he has to show the slips to pass through, en route to his destination.

Some cities have enforced lockdown, that means those cities have blocked the entry into the cities and you will have to skip the cities in any case, such cities are represented by -1. This information is known to Suresh upfront.

Help Suresh find the minimum amount to be paid to reach from City A to City B

Constraints

No of cities <=10^5

Input

First line contains an integer N, denoting the number of cities

Second line contains N space separated integers, where first integer denotes the cost of issuing itinerary slips and next (N-1) integers denote the entry fee of all cities. The last integer is always the destination city. If city is under lock down then its entry fee will be -1.

Third line contains an integer, M which represents number of cities he can skip from a present city during his travel

Output

Single integer which represents the minimum cost Suresh has to pay to travel from city A to B, but if city B is not reachable then print -1

Time Limit

Examples

Example 1

Input

5

1 6 -1 5 7

1

Output

19

Explanation

Since he could skip only 1 city between the cities. He will have to pay 1+6+5+7, where 1 is the fees paid to issue slips and [6,5,7] are the fees paid for the entry to the respective cities. So the total amount he has to pay while leaving A is 19.

Example 2

Input

4

3 4 1 -1

3

Output

-1

Explanation

Since the city B is under lockdown, he cannot go to city B. Hence the output is -1