5^{th} Virginia Tech High School Programming Contest

Dec 8, 2018

As a reminder, here are the key rules under which this contest is conducted:

- Teams may not communicate with another human during the contest about the problems.
- Teams may not use more than 1 computer.

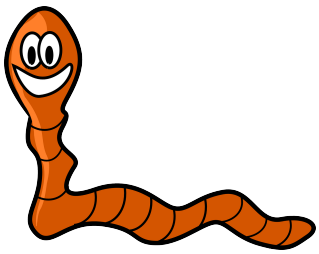
This problem set contains a large number of problems (13) which target a variety of skill levels. You are not expected to solve all of them, particularly if this is your first programming contest!

Enjoy!

This page is intentionally left blank.

Problem A
Climbing Worm

A worm is at the bottom of a pole. It wants to reach the top, but it is too lazy to climb to the top without stopping. It can crawl up the pole a certain number of inches at a time, falling down a lesser number of inches right after while it rests. How many times does the worm need to crawl up in order to reach the top of the pole?



Source: Pixabay

Input

The input consists of a single line that contains three integers a, b ($0 \leq b < a \leq 100$), and h, ($0 < h \leq 100\,000$) indicating the amount a of inches the worm can climb at a time, the amount b of inches the worm falls during its resting period, and the height h of the pole, respectively. Note: For the purposes of this problem, the worm is modeled as a point and thus has no length.

Output

Output the number of times the worm must crawl up in order to reach the top of the pole.

Sample Input 1	Sample Output 1
5 0 15	3

Sample Input 2	Sample Output 2
3 1 4	2

This page is intentionally left blank.

Problem B
Pedal Power

Photo courtesy of Christy Coghlan

You're ready to start your semester off right! Well, almost... you have everything listed on your calendar, you just bought a brand new bike, but you haven't yet planned your commute! You are quite busy this semester, and so you decide to plan a route for all your classes and appointments which requires the least amount of travel time. You know how long certain routes will take with and without a bike. You can choose to ride or not ride your bike at any point, but if you leave your bike somewhere you must return to the same location to pick it up before riding the bike again. Additionally, you can't carry your bike on non-bike routes. What is the shortest amount of cumulative time it will take to travel to all the places you need to go and return you and the bike to your starting location?

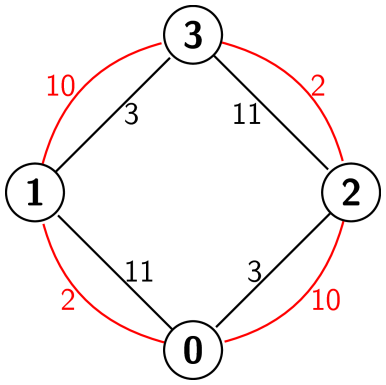


Figure B.1: This figure shows sample input 1, with bike paths in red/curved and non-bike paths in black/straight.

Input

The problem input describes a multigraph with two different types of edges: edges for bike paths and edges for non-bike paths.

The first line contains one integer n, $0 < n \leq 300$, the number of locations on campus. The locations will be numbered from 0 to $n-1$. Location 0 is your home, which is the starting and ending location of you and your bike.

The second line contains one integer x, $0 < x \leq n(n-1)/2$, the number of bike paths.

Each of the next x lines contains a description of a bike path. Each line will have 3 integers u v t, where $0 \leq u < n$ is the starting location of the path, $0 \leq v < n$ is the ending location of the path, and $0 \leq t \leq 10^6$ is the time it takes to travel from u to v and from v to u by bike.

The next line contains one integer y, $0 < y \leq n(n-1)/2$, the number of non-bike paths.

Each of the next y lines contains a description of a non-bike path. Each line will have 3 integers $u\ v\ t$, where $0 \leq u < n$ is the index of the starting location of the path, $0 \leq v < n$ is the index of the ending location of the path, and $t \leq 10^6$ is the time it takes to travel from u to v and from v to u using a bike alternative.

All paths can be taken in either direction, and the amount of time it takes to travel on either direction of the same path is the same. There will be at most one bike path and at most one non-bike path between any two locations. No path starts and ends at the same location.

The next line contains an integer z, $0 < z \leq 300$, indicating the number of locations you have to visit on your route. The next line will contain z integers $0 \leq a_1, a_2, ..., a_z < n$, the indices of the locations you must visit. The locations must be visited in the order in which they are listed, but there are no restrictions on how often you may pass through any location on your trip.

Output

Output the minimum possible cumulative travel time, with you and the bike starting and ending at location 0. You are guaranteed that all locations on your route can be reached using some combination of bike and non-bike paths.

Sample Input 1	Sample Output 1
4 4 0 1 2 3 1 10 2 3 2 2 0 10 4 1 0 11 3 1 3 2 3 11 2 0 3 3 1 3 2	16

Problem C
Math Homework

Source: pixabay

Since entering 2nd grade Theta has daily math homework sheets. The problems on her worksheet usually go like this:

> There is a certain number of birds, dogs, and cats on a farm. Together they have 14 legs. How many birds, dogs, and cats could there be? Write down as many answers as you can!

It is always the same problem, just written in different ways: sometimes with horses, cows, sheep, goats, chickens, beetles, or even spiders (but never with snakes or fishes!)

Can you write a program to double-check Theta's answers?

Input

Input consists of a single line with 4 integers: b, d, c, and l, with b, d, and c representing the numbers of legs the first, second, and third type of animal has. You are given that $0 < b, c, d \leq 100$ because some farm animals in these math problems may be centipedes. The total number of legs is given by l ($0 \leq l \leq 250$).

Output

Output all possible answers, each on a separate line, in lexicographical order so that they are sorted by the number of the first animal, ties broken by the second and third animal, respectively. Separate the number of the first, second, and third animal with spaces. If there are no possible solutions, output `impossible` on a single line!

Sample Input 1	Sample Output 1
2 4 4 14	1 0 3 1 1 2 1 2 1 1 3 0 3 0 2 3 1 1 3 2 0 5 0 1 5 1 0 7 0 0

Sample Input 2	Sample Output 2
100 80 60 240	0 0 4 0 3 0 1 1 1

Sample Input 3	Sample Output 3
2 4 6 9	impossible

Problem D
Pharmacy

Source: pixabay

Many pharmacies in the United States fill prescriptions strictly on a first-come, first-served basis, even preferring prescriptions submitted electronically from a remote site to prescriptions dropped off by waiting customers in the store. This frequently leads to situations where customers have to wait for prescriptions because pharmacy technicians are busy filling prescriptions that may not be picked up until much later anyway.

This problem asks you to investigate the effect of an "in-store customers first" policy. Under this policy, technicians fill prescriptions in the order in which they are dropped off, but they will not start filling remote prescriptions as long as there are in-store prescriptions to be filled.

Write a program that computes the average completion time for in-store and remote customers under this policy!

Input

The input consists of a single test case. The first line contains two integers n ($0 < n \leq 100\,000$) and t ($0 < T \leq 10$), where n denotes the number of prescriptions and T the number of technicians working at the pharmacy. This is followed by n lines, one line per prescription.

Each line consists of one integer d ($0 < d \leq 10^9$), which is the time (in seconds) when the prescription was dropped off, followed by a single character '`R`' or '`S`' for a remote or in-store prescription, followed by a single integer k ($0 < k \leq 500$) describing the number of seconds it takes a technician to fill this prescription. Prescriptions are listed in increasing order of drop-off time, but multiple prescriptions may be dropped off at the same time, in which case preference should be given to any in-store prescriptions, regardless of the order in which they appear in the input. If two prescriptions of the same type are dropped off at the same time, the prescription needing the shorter fill time must be filled first, again regardless of the order in which they appear in the input.

Once a technician starts working on a prescription, they will finish filling it, even if an in-store prescription is dropped off while they are working. Each prescription requires one pharmacy technician to fill it, and each technician can work on only one prescription at a time. The technicians work until all prescriptions are filled. You should ignore the time it would take for technicians to decide which prescription to fill next, so a technician that finishes a prescription can start working immediately on the next prescription.

Output

Output two numbers o and r, denoting the average completion time for in-store and remote prescriptions, respectively. A prescription's completion time is defined as the time difference between when a prescription was dropped off and when it was filled. If there were no in-store or remote prescriptions in the input, output 0 for o or r instead. Your answer is considered correct if it is within a relative or absolute error of 10^{-6}.

Sample Input 1

```
5 3
1 R 4
2 R 2
3 R 2
4 S 2
5 S 1
```

Sample Output 1

```
1.500000 2.666667
```

Sample Input 2

```
5 2
1 R 4
2 R 2
3 R 2
4 S 2
5 S 1
```

Sample Output 2

```
1.500000 3.666667
```

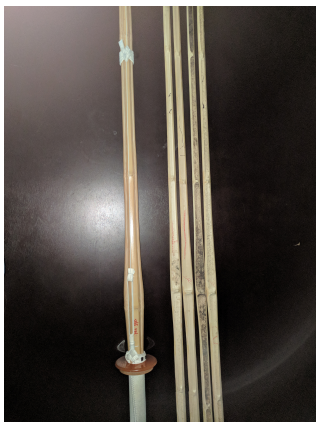
Sample Input 3

```
5 1
1 R 4
2 R 2
3 R 2
4 S 2
5 S 1
```

Sample Output 3

```
3.000000 7.000000
```

Problem E
Broken Swords



Source: Dan Folescu

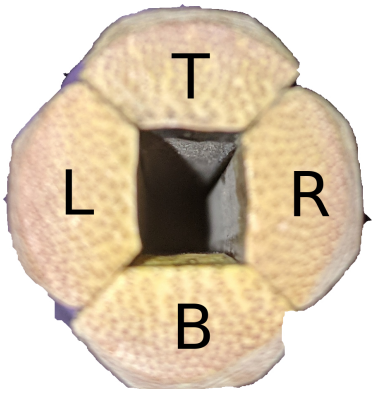
Ken is a fencer with a big problem: he swings too hard!

Whenever Ken gets a new sword, he's sure to break it sooner rather than later. It's occurred to him that this habit is costing him quite a bit of money, since he normally buys a new sword whenever his current sword breaks. However, he's heard from a fellow club member that he could instead make new swords out of his old swords!

Each bamboo sword Ken uses has four bamboo 'slats' which help to cushion each blow for the receiving partner. Whenever Ken breaks his sword, he breaks either one, two, three, or all four slats.

When Ken first tried to put a sword together with the remains of two broken swords, he realized that not all slats are equivalent. When the slats are bundled together, they conform to the shape of the slat which sits opposite!

Ken doesn't want an uncomfortable grip, so he will put only an old slat in a new position if it is of the same or opposite type. There are four possible slats "top" (`T`), "bottom" (`B`), "left" (`L`), and "right" (`R`). When looking at a sword head on, these slats form the following configuration:



As an example, Ken will only replace a top slat with a top or bottom slat.

After collecting broken swords for quite a while, Ken has decided to build as many swords as possible. Write a program to compute how many swords he will be able to build and how many slats he'll have left over!

Input

The input consists of a single test case. On the first line, you are given N, the number of swords Ken has broken. On each of the next N lines, you are given four characters. The characters correspond to the following slat ordering: `TBLR`. If the character in a position is `0`, then the slat is not broken, and if it is a `1`, Ken is out of luck!

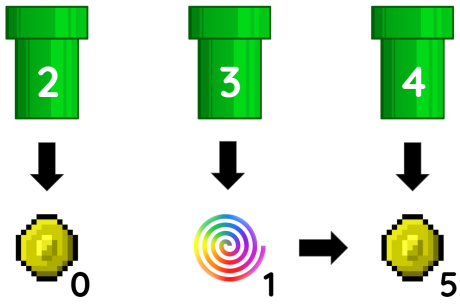
Output

Output three numbers: the total number of swords Ken can make, the sum of the numbers of the remaining T and B slats, and the sum of the numbers of the remaining L and R slats, respectively.

Sample Input 1	Sample Output 1
4 0100 0010 0110 1010	2 1 1

Problem F
Bowser's Pipes

Princess Peach is in trouble again! Of course, Mario is on a quest to rescue her from Bowser. To save her, he has to collect a number of coins located in coin rooms under the surface. To get to a coin room, he must enter pipes on the surface which send him to a room below.



This illustration matches sample input 1.

Unfortunately, some pipes don't lead directly to a coin room, but lead to a warp room instead: those warp rooms might lead to other warp rooms in turn, until Mario eventually reaches a coin room. Once Mario chooses a pipe to enter, he'll be warped through the pipes without regaining control until he reaches a coin room.

Mario doesn't like getting warped and because of this, he is asking his brother Luigi for help. He's talked Luigi into jumping down some of the pipes to find out to which coin room they ultimately lead and share this information with him.

Mario then will show off his best rescue skills by jumping down the pipe that leads to the same coin room his brother Luigi found, but which passes the smallest possible number of warp rooms on the way, thus saving Mario from nausea and impressing Princess Peach!

Input

The first line of input contains an integer N ($0 < N \leq 100\,000$), the combined number of pipes, warp rooms, and coin rooms, which are numbered from $0 \ldots N-1$. The second line contains N space-separated values F_i ($-1 \leq F_i < N$) where F_i indicates the number of the warp or coin room to which Mario is propelled when he enters the pipe or the warp room with number i. $F_i = -1$ indicates that Mario has finally reached a coin room. Each warp room has at least one pipe leading to it. You are guaranteed that Luigi will reach a coin room after a finite number of steps, no matter which pipe he jumps into.

The third line of input contains an integer A ($0 < A \leq 100\,000$), the number of pipes Mario asks Luigi to jump into. The following line contains A numbers D_i ($0 \leq D_i < N$), each denoting the number of a pipe that Luigi enters.

Output

For each pipe that Luigi enters, find out to which coin room it leads, and then print the number of the pipe Mario should enter to impress Princess Peach. This is the pipe that leads to the same coin room, but with the smallest number of warp rooms on the way. In the case of a tie, output the pipe with the smallest number!

Sample Input 1	Sample Output 1
6 -1 5 0 1 5 -1 2 2 3	2 4

Problem G
Touchdown!

Source: pixabay

In a game of American Football, there are several factors to take into consideration when detailing the rules of the game. The goal of the game is to move the football across the field into the endzone of the defending team through a series of plays, which form a drive. In each play, the attacking team may either gain or lose a number of yards, that is, their position on the field moves either towards or away from the defending team's endzone.

In the simplified version of football that we will consider, a team has four chances (in up to four plays) to advance the ball 10 yards downfield; if they are successful, they will have achieved a "first down" and keep possession of the ball. If they achieve a first down, they have another up to four plays to continue their drive towards the defending team's endzone. If they keep possession and reach the defending team's endzone, they will have achieved a Touchdown. If they are pushed back into their own endzone, a Safety occurs and ends the drive. Otherwise the team loses possession of the ball and Nothing happens.

Sadly, the outcome of a drive has been lost, and all that remains is the yards gained or lost on each play! Your job is to determine whether or not a Touchdown, Safety, or Nothing occurred on the given drive.

For simplicity, we will assume that the team starts the drive on their own 20 yard line on a 100 yard field (with 0 being the team's own endzone, and 100 being the defending team's endzone). This means that a touchdown is scored if at least 80 yards are gained in total, relative to the starting position, and without losing possession due to failing to get a first down. (If a team gains 80 yards on the first play, they will have achieved a Touchdown even though they didn't achieve a first down in between.) A safety occurs if the team is pushed back 20 yards from their original starting position, which would place them in their own endzone. Nothing occurs if neither of these events occurs.

Input

The input consists of a line containing one integer N ($1 \leq N \leq 15$), which is the number of plays that this given drive recorded. Following this line are N numbers representing the numbers of yards gained or lost on each particular play. Each given number will be between -100 and 100 yards since that is the length of the football field.

Output

Output a single word, the result of the drive! If a touchdown is achieved, output "`Touchdown`", if a safety is achieved, output "`Safety`", else output "`Nothing`". (Do not add a period at the end.) Once the outcome has been determined, your program should ignore the remaining yards listed in the drive.

Sample Input 1	Sample Output 1
9 10 3 8 22 -4 16 8 3 14	Touchdown

Sample Input 2	Sample Output 2
10 9 15 2 -5 3 8 18 3 25 2	Nothing

Problem H
Substitution Mania!

Source: pixabay

Your friend Ben absolutely loves substitution ciphers!

A substitution cipher takes the alphabet and lines up a substitution with it, as shown in the example below:

```
abcdefghijklmnopqrstuvwxyz
saucpwhkjdzxgitfmvnyeborql

unencrypted -> encrypted
a -> s
b -> a
...
z -> l
```

Then, for each letter of the plaintext, we find that letter in the top row and substitute it with the corresponding letter in the bottom row. The cipher is referred to by the string in the bottom row. In this example, "hello abc" would encrypt to "kpxxt sau" (Ben leaves spaces as spaces so things don't get too confusing).

Ben always encrypts his messages to you using between 1 and 12 substitution ciphers which he applies consecutively because he doesn't want anyone else getting into your business.

He also gives you the substitution ciphers he uses, but to increase security he doesn't give them to you at the same time. Unfortunately, sometimes he ends up giving them to you in the wrong order!

Luckily, you have the plaintext and ciphertext of another message that used exactly the same sequence of ciphers. Given this message and the list of ciphers, write a program to decrypt Ben's latest message!

Input

The first line contains d_0, the plaintext of the message you have, with length$(d_0) \leq 10\,000$. The second line contains e_0, the ciphertext that corresponds to d_0.

The third line contains an integer $1 \leq i \leq 12$, the number of substitution ciphers Ben used.

The following i lines each contain one of the substitution ciphers, each of which is a nonrepeating string of exactly 26 lowercase letters.

The last line contains e_1, the ciphertext of the message you should decrypt.

All lines containing text consist of only lowercase English letters and spaces.

You are guaranteed that there is a unique sequence of applications of the given substitution ciphers that encrypts d_0 to e_0 in which each of the given ciphers is used exactly once.

Output

Output the plaintext d_1 obtained after decrypting the ciphertext e_1 on a single line.

Sample Input 1

```
hey check out this cool substitution cipher
kpq ukpuz tey ykjn uttx neanyjyeyjti ujfkpv
1
saucpwhkjdzxgitfmvnyeborql
fxpsnp vpnftic
```

Sample Output 1

```
please respond
```

Sample Input 2

```
hey its your friend ben
lik grp kfnm zmgiyj ciy
5
orcfnushpmbyxzjltdviqgaewk
cwnuayzsqekgltdprovmbhxfji
yfizduprnchxkbvalgwmejoqst
aoxenqjzclgwkytrpdbmuvihfs
tyjgklxocquemhanirdwszvpbf
thy kfn mihj rlgp
```

Sample Output 2

```
can you read this
```

Problem I
The Grand Adventure

Our hero, Jim, is about to embark on his grand adventure. On his way, he'll encounter three different types of objects: money, incense, and gems. When he encounters an object, he will always put it into his backpack. Unfortunately, he'll also encounter villains along the way. More specifically, the Banker (who will demand money), the Trader (who demand incense), and the Jeweler (who will demand, of course, jewels). Jim must give each villain one of the kind of item they demand as he encounters them, or else he fails and his adventure is over. Unfortunately, Jim's backpack isn't very ergonomic and so he can only reach the item he most recently put in it. In other words, the items below are inaccessible until he's given away the one on top. Jim would also like arrive at his destination with no extra items in his bag. If he does have items at the end, he'll consider his adventure a failure.

We'd like to know if Jim will be able to complete his adventure before he even starts!

Input

The first line contains a single integer n, $1 \leq n \leq 5$, the number of adventures. Each of the next n lines contains a single string a, a sequence of $\{\$, *, |, t, j, b\}$. An example of this adventure string is:

```
....$...$...*..*...|...t...j..j...b..b...
```

where,

- `$` represents Money
- `|` represents Incense
- `*` represents Gem
- `t` represents a Trader
- `j` represents a Jeweler
- `b` represents a Banker
- `.` represents the Ground (nothing)

and $1 \leq |a| \leq 100$ where $|a|$ is the length of string a. There is no limit on the number of items Jim may place in his backpack.

Output

Print `YES` if Jim is able to finish his adventure and print `NO` if he is unable to.

Sample Input 1

```
5
.$.b.|.t.*.j................$$$$$bbbbb||....tt.....
.$.b.|.t.*.j..
..............................
.......$....$......*....*.....|......t........j...j.....b..b........
...$.$.$..*..*..*...*..|..*..b.....********...
```

Sample Output 1

```
YES
YES
YES
YES
NO
```

Sample Input 2

```
4
........$b...$$..t...*..*...j.........j...
.........*****jjjj..............|tj....
.$.|.*.$.|.*.$.|.*.j.t.b.j.t.b.j.t.b.
...$$..$$..$$..|..$$..b......b....t..bbbbbb.....j...
```

Sample Output 2

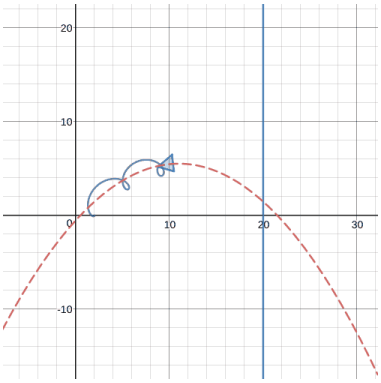
```
NO
YES
YES
NO
```

Problem J
Holiday Stars

It is the time for the Holidays, which means it's time for the annual Holiday Cookie Throwing contest where cookies are thrown against a wall. Cookies are really simple polygons that rotate clockwise while in flight.

Due to many years of advanced research and precise throwing equipment, these cookies can fly under idealized conditions: their center of mass follows an ideal parabola ($g = 9.81\frac{m}{s^2}$), and their angular velocity stays constant.

Given a cookie's shape, initial angle and speed as well as its angular velocity, compute which of the cookie's corners hits the wall first, and when!



A Holiday Star hurtling towards a wall

Input

The input consists of a single test case. The first line contains 5 numbers: an integer n ($3 \leq n \leq 20$) denoting the number of vertices of the cookie polygon, a real number ω ($0 \leq \omega \leq 10$), the angular velocity in rad$/s$ in *clockwise* direction, a real number v_0 ($1 < v_0 < 10$), the initial velocity of the cookie in $\frac{m}{s}$, a real number θ ($0 \leq \theta \leq 80$), the initial angle of the cookie's trajectory, given in degrees measured counter-clockwise relative to the $(1, 0)$ unit vector, a real number w ($20 \leq w \leq 500$), denoting the x-coordinate of the wall towards which the cookie is moving. You may assume that there are no other objects in space aside from the cookie, and that the wall is of infinite height (e.g., from $-\infty \ldots \infty$).

This is followed by n lines, one for each vertex. Each line contains two real numbers x_i and y_i ($|x_i| < 20, |y_i| < 20$) denoting the initial position of a cookie's corner. Corners are numbered $1 \ldots n$. The cookie polygon's vertices are given in counter-clockwise order. The polygon is simple (e.g., not self-intersecting) and has a non-zero area.

All real numbers are given with no more than 6 decimal digits after the period.

Output

Output two numbers i and T_i, denoting the index i ($1 \leq i \leq n$) of the corner that hits the wall first and the time T_i in seconds when that happens. The index corresponds to the order in the input. T_i must be given with an absolute error of less than 10^{-3}! You may assume that no two vertices hit the wall within 10^{-2} seconds of each other, and that the first vertex to hit the wall would pierce it for at least 10^{-3} seconds.

Sample Input 1	Sample Output 1
3 6 5 45 20 0 0 2 0 1 1.5	2 5.086781

Sample Input 2

```
3 0.25 2 45 20
0 0
2 0
1 1.5
```

Sample Output 2

```
1 12.715255
```

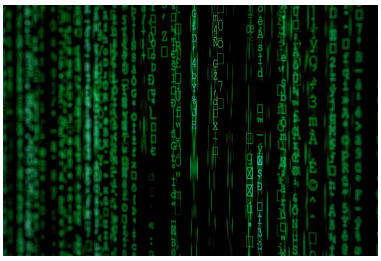
Sample Input 3

```
3 0 2 0 20
0 0
2 0
1 1.5
```

Sample Output 3

```
2 9.000000
```

Problem K
Good Messages


Source: pexels.com

Boris works at a secret communication station for the government training new employees on how to encode messages. Messages at this station are encoded with a rotation ("Caesar") cipher that replaces each letter with one at a set offset in the alphabet, e.g., for offset 2 an 'a' is replaced by a 'c' and 'y' is replaced by 'a'. In order to encode a message, this cipher may be applied more than once. Each such application counts as a step in the encoding process. Boris teaches the new employees to encode messages by demonstrating each step of the encoding individually. However, Boris does not does not like seeing messages that contain at least half as many vowels as consonants after applying an encoding step. (Boris considers 'a', 'e', 'i', 'o', 'u', and 'y' as vowels). He grows annoyed and more and more unhappy with each step where this situation occurs.

Your job is to encode a given message and determine whether Boris will see fewer steps that annoy him than ones that don't. Since Boris wants to be happy in his job he will give a message that annoys him too much to a colleague.

Input

The input consists of a single test case. The first line of the input contains a single integer O ($1 \leq O \leq 25$) that represent the offset used for the rotation cipher. The second line contains the message to encode, which consists entirely of lowercase English characters. The length of the message is between 1 and 80 characters. The third line will contain an integer N ($1 \leq N \leq 26$), the number of times the cipher must be applied.

Output

Output 'Boris' if strictly more steps sound good than bad, and 'Colleague' otherwise.

Sample Input 1

```
1
thequickbrownfoxjumpedoverthelazydog
10
```

Sample Output 1

```
Boris
```

Sample Input 2

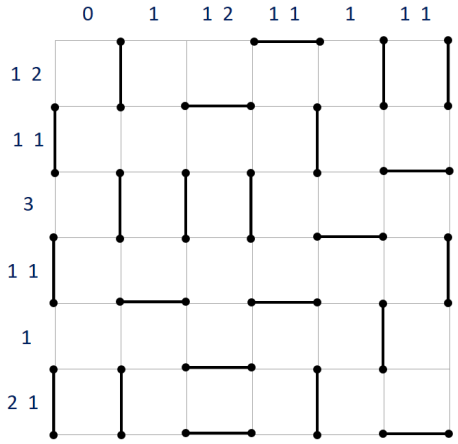
```
4
banana
3
```

Sample Output 2

```
Colleague
```

This page is intentionally left blank.

Problem L
Bar Code



Bar Code is a puzzle game invented by Thinh Van Duc Lai in 2017 that was first published in the New York Times. Bar Code is played on a square grid consisting of $n \times n$ unit squares with $(n+1) \times (n+1)$ grid points. For a given specification, the puzzle requires the marking of some of the borders of these unit squares by drawing a vertical or horizontal bar, subject to the following conditions:

- For each row/column, the specification describes exactly how many separate groups of consecutive bars of the same orientation (vertical and horizontal, resp.) there should be in this row/column. For instance, if the specification demands `2 1 1`, then there have to be 3 groups of 2, 1, and 1 bars, separated by at least one unmarked border. For example, for a $n = 6$ puzzle with 7 borders in each row, the following markings would meet this specification:

  ```
  1101010
  1101001
  1100101
  0110101
  ```

  where `1` denotes the presence of a bar and `0` denotes a border that is unmarked.

- No 2 bars may touch.

The illustration shows the solution for sample input 3.

Find a marking that is consistent with the given specification!

Input

The input consists of a single test case. The first line contains a single integer n ($0 < n \leq 9$). This is followed by n lines. The i^{th} line contains 1 or more non-negative integers denoting the sizes of groups that must be formed by the vertical bars in row i (counting from the top). If the line contains the single integer 0, then no borders may be marked for that row. Otherwise, none of the integers on the line will be 0.

Following that will be another n lines. The j^{th} line contains 1 or more non-negative integers denoting the sizes of groups that must be formed by the horizontal bars in column j (counting from the left). If the line contains the single integer 0, then no borders may be marked for that column. Otherwise, none of the integers on the line will be 0.

Output

Output the solution as follows. On the first n lines, output a string of length $n + 1$ consisting of `1` and `0` characters. The i^{th} string should contain a `1` in position j if and only if the j^{th} vertical border in row i should be marked as a bar. On the next $n+1$ lines, output a string of length n also consisting of `1` and `0` characters. The i^{th} string should contain a `1` in position j if and only if the i^{th} horizontal border in column j should be marked as a bar. Rows are counted top down and columns are counted left to right.

If there are multiple solutions, you may output any of them! You may assume that at least one marking exists that is consistent with the specification.

Sample Input 1

```
2
1
0
0
3
```

Sample Output 1

```
100
000
01
01
01
```

Sample Input 2

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

Sample Output 2

```
0000
1001
0010
101
010
000
100
```

Sample Input 3

```
6
1 2
1 1
3
1 1
1
2 1
0
1
1 2
1 1
1
1 1
```

Sample Output 3

```
0100011
1000100
0111000
1000001
0000010
1100100
000100
001000
000001
000010
010100
001000
001001
```

Problem M
Competitive Arcade Basketball

Source: pixabay

You're attending a arcade basketball competition, where the objective is to score as many points as possible until the time runs out. The announcer has informed the crowd that their scoreboard is broken, so they don't have a way to keep track of all the scores. As a seasoned programmer, you feel you can whip up a program that can keep track of the names of the players and the amount of points they've scored, announcing the winner(s) at the end of the contest.

Input

The first line contains three integers: the number of participants n ($1 \leq n \leq 100\,000$); the minimum number p of points required to win the contest ($10 \leq p \leq 10\,001$); and m, the number of lines with player names and points ($1 \leq m \leq 200\,000$). The next n lines contain the names of the participants, each mentioned exactly once. Each name consist of no more than 20 alphanumerical characters. The remaining m lines each contain the name of a participant, followed by how many points they scored (1, 2, or 3).

Output

Output the names of those participants who reached the minimum required score, one per line! Output "`<Winner> wins!`" for each winner. Output the winners in the order in which they've reached the required score. If no one reaches the minimum required score, output "`No winner!`" (including the exclamation mark!).

Sample Input 1

```
3 10 13
John
Kelly
George
Kelly 1
George 2
Kelly 1
John 2
George 1
John 3
Kelly 3
Kelly 1
George 3
George 1
John 3
George 3
Kelly 1
```

Sample Output 1

```
George wins!
```

Sample Input 2

```
4 10 13
Bob
Nina
Jess
Tim
Nina 2
Bob 2
Nina 1
Jess 3
Bob 2
Jess 2
Nina 1
Jess 2
Nina 3
Bob 1
Nina 3
Jess 3
Bob 2
```

Sample Output 2

```
Nina wins!
Jess wins!
```

Sample Input 3	Sample Output 3
4 15 13 Bob Nina Jess Tim Nina 2 Bob 2 Nina 1 Jess 3 Bob 2 Jess 2 Nina 1 Jess 2 Nina 3 Bob 1 Nina 3 Jess 3 Bob 2	No winner!

This page is intentionally left blank.