MAPS Python Challenge 2014

Problem set

BEKK

Oslo, 13.03.2014

Table of contents

Problems

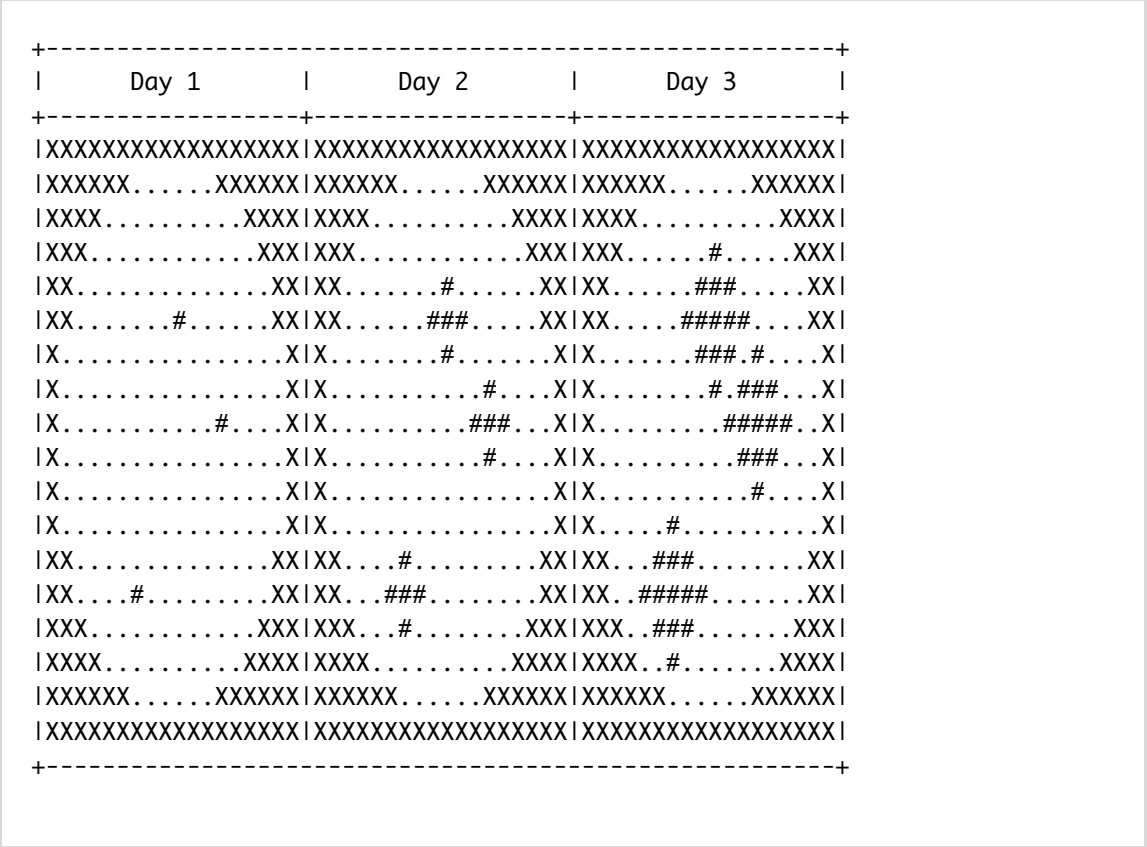
Sample problems with solution

Bacterial growth

Lisa is a lab technician working at a hospital testing blood samples. Testing blood samples can be a labour intensive undertaking. For each sample Lisa must first prepare a glass plate with a growth medium, then add some of the blood from the sample. After a day has passed Lisa counts the number of visible bacteria cells, using a microscope. Then after another couple of days Lisa recounts the number of bacteria. The amount of growth in the given time span gives the doctors an impression of how aggressive the bacteria is.

Lately Lisa has been testing a lot of samples for the *Quatuor* bacteria, which she has noticed grows in a very predictable way. If the glass plate is divided into a grid where each bacteria cell occupies a grid cell, the growth during one day can be described as follows: Each cell divides into five, such that there is a cell added north, south, east and west of the original cell. The only exceptions from this growth pattern is when any of these grid cells are already occupied of other cells, doesn't contain growth medium or are outside of the glass plate.

An example image series of such growth can be seen below. `X` indicates an empty grid cell, `.` indicates a grid cell with growth medium and `#` indicates a grid cell with a bacteria cell in it.



Instead of wasting time on performing tests she can predict the result for, Lisa has better things to do. Therefore she needs you to make a program that can predict the result of the test, given an image of the glass plate after day 1. To make sure her boss doesn't discover that she's not actually performing the tests, Lisa will vary the size of the glass plates, and the number of days between the initial and the final count.

Input specification

The first line of input contains three integers *m*, *n*, the number of rows and columns in the glass plate grid, and *t*, the number of days the bacteria should grow before the final count. Then follows *m* lines, each containing *n* characters. These lines make up the grid image of the glass plate. The characters `#`, `.` and `X` represent a grid cell containing a bacteria cell, a grid cell with growth medium and an empty grid cell, respectively.

Output specification

Output the number of bacteria cells there are on the glass plate after *t* days of growth, given the initial state in the input.

Constraints

$1 \leq m, n \leq 100$
$1 \leq t \leq 20$

Sample input 1

```
18 18 2
XXXXXXXXXXXXXXXXXX
XXXXXX......XXXXXX
XXXX..........XXXX
XXX............XXX
XX..............XX
XX.......#......XX
X................X
X................X
X...........#....X
X................X
X................X
X................X
XX..............XX
XX....#.........XX
XXX............XXX
XXXX..........XXXX
XXXXXX......XXXXXX
XXXXXXXXXXXXXXXXXX
```

Sample output 1

```
39
```

Sample input 2

```
14 14 2
XXXXXXXXXXXXXX
.........XXXXX
.........XXXXX
.........XXXXX
....#....X...X
.........X.#.X
.........X...X
.........XXXXX
.........XXXXX
....#....XXXXX
....#....XXXXX
.........XXXXX
.........XXXXX
.........XXXXX
```

Sample output 2

```
40
```

Beer shopper [easy]

Greg is a good guy, who likes to look after his friends. Being a good guy means Greg has to buy beers for his friends all the time. As we all know beer comes in sixpacks, so Greg has to buy beers in multiples of six. Greg wants to buy exactly one beer for each of his friends. This is because if someone gets more than one, there will be a fight. Needless to say, leaving one or more beers undrunken is out of the question.

Help Greg out writing a program that determines whether it is possible to buy beer in such a way that each of his *f* friends get exactly one.

Input

Input consists of a single line with a single integer *f*, the number of friends Greg wants to buy beer to.

Output

If it is possible to buy a number of sixpacks such that there is exactly one beer for each of Greg's friends, simply output `BEER!`. Otherwise output `FIGHT!`.

Constraints

$0 < f \leq 10^9$

Sample input 1

```
6
```

Sample output 1

```
BEER!
```

Sample input 2

```
12
```

Sample output 2

```
BEER!
```

Sample input 3

```
15
```

Sample output 3

```
FIGHT!
```

Beer shopper 2

(See *Beer shopper* for full introduction)

It turns out giving away beer for free is a great way to get new friends! Greg has made so many new friends he has trouble keeping up with demand.

Luckily he has a friend working in a brewery, who has come up with a deal where Greg can buy beer in bulk. The beer from the brewery comes in boxes with `198` beers.

As before Greg wants to share exactly one beer with each of his friends, without having any excess beer (which will lead to fighting). Write a program that determines whether it is possible to buy beer from the brewery in such a way that each of his *f* friends get exactly one.

Note that this time Greg has got *really* many friends.

Input

Input consists of a single line with a single integer *f*, the number of friends Greg wants to buy beer to.

Output

If it is possible to buy a number of boxes such that there is exactly one beer for each of Greg's friends, simply output `BEER!`. Otherwise output `FIGHT!`.

Constraints

$0 < f \leq 10^{10^6}$

Sample input 1

```
100
```

Sample output 1

```
FIGHT!
```

Sample input 2

```
198
```

Sample output 2

```
BEER!
```

Sample input 3

```
297
```

Sample output 3

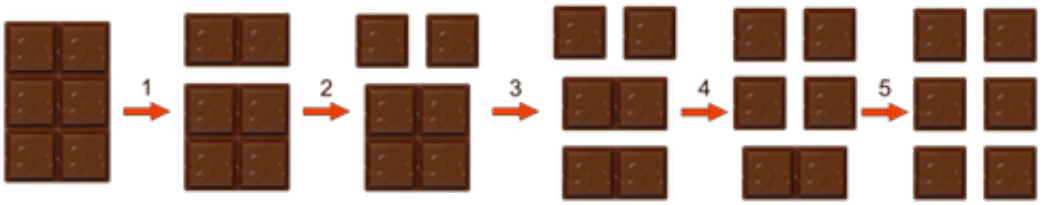
```
FIGHT!
```

Chocolate

Sverre loves chocolate, and he eats a lot of it. Actually, the only thing he eats is chocolate. Since chocolate is such a big part of Sverre's life, he naturally spends much time on it, and the most time consuming part is of course the chocolate bar splitting. Therefore he asks you to help him optimizing the process.

The original chocolate bar consists of *R* rows with *C* pieces in each row. Sverre has to divide it into *R* * *C* pieces by doing splits.

A split is a simple process. Sverre picks one of the existing chocolate bars with more than one piece, and divides it into two new chocolate bars by cutting along one of the separator lines. The split process is done until all the pieces are separated. The image below shows how to separate all the pieces in a 3x2 chocolate bar with 5 splits.



Given *R* and *C*, what is the least number of splits needed to turn the original chocolate bar into *R* * *C* pieces.

Input:

The input consists of a single line containing two integers: *R* and *C*.

R is the number of rows in the chocolate bar. *C* is the number of pieces in each row.

Output:

Output on a single line, one number, the least number of splits needed to divide the original chocolate bar into *R* * *C* pieces.

Constraints

$1 \leq R \leq 10^4$
$1 \leq C \leq 10^4$

Sample input 1:

```
3 2
```

Sample output 1:

```
5
```

Sample input 2:

```
4 4
```

Sample output 2:

```
15
```

Election day 1

Note:

This is one of two problems named "Election day". These problems (i.e. "Election day 1" and "Election day 2") differ only by their input constraints. See section "Constraints".

Problem text

Tomorrow is election day at MAPS and the time has come to choose a new leader for the organization. There is, however, one problem: how to pick a leader when every reasonable person wants to be the new leader? In such a case, every person at the election would simply vote for herself.

Yang (the mathematician) suggests an easy process to pick the new leader. It goes as follows: the *N* candidates will place themselves at positions 1 through *N* on a line. Then every second person (starting at the second position) is consecutively eliminated. The eliminated candidates then leave the line with the remaining candidates taking positions 1, 2, ... If the last person on the previous line was eliminated, the process restarts as before on the second person. If not the process restarts on the first person. This continues until there is only one candidate left.

For example, if *N*=5 the candidate's initial positions are 1,2,3,4 and 5. The first candidates to be eliminated are at positions 2 and 4. They leave the line. Then candidate at position 1 stays, the candidate at 3 goes to position 2 and the candidate at 5 goes to position 3, so the new line will consist of positions 1,2 and 3. The process then restarts at position 1. The candidates at positions 1 and 3 are now eliminated. There is only one candidate remaining, on position 2. This is the new leader.

The board is extremely happy they now have a way to choose a new leader!

Little do the board know of Yang's wicked plans. He will calculate the position of the new

leader on the initial line and place himself there.

Input specification

The input consists of a single number N on a single line, the number of candidates showing up.

Output specification

Output on a single line, one number, i, Yang's position on the initial line.

Constraints

$1 \leq N \leq 10^5$

$1 \leq i \leq N$.

Sample input 1

```
5
```

Sample output 1

```
3
```

Sample input 2

```
1
```

Sample output 2

```
1
```

Election day 2

Note:

This is one of two problems named "Election day". These problems (i.e. "Election day 1" and "Election day 2") differ only by their input constraints. See section "Constraints".

Problem text

Tomorrow is election day at MAPS and the time has come to choose a new leader for the organization. There is, however, one problem: how to pick a leader when every reasonable person wants to be the new leader? In such a case, every person at the election would simply vote for herself.

Yang (the mathematician) suggests an easy process to pick the new leader. It goes as follows: the *N* candidates will place themselves at positions 1 through *N* on a line. Then every second person (starting at the second position) is consecutively eliminated. The eliminated candidates then leave the line with the remaining candidates taking positions 1, 2, ... If the last person on the previous line was eliminated, the process restarts as before on the second person. If not the process restarts on the first person. This continues until there is only one candidate left.

For example, if *N*=5 the candidate's initial positions are 1,2,3,4 and 5. The first candidates to be eliminated are at positions 2 and 4. They leave the line. Then candidate at position 1 stays, the candidate at 3 goes to position 2 and candidate 5 goes to position 3, so the new line will consist of positions 1,2 and 3. The process then restarts at position 1. The candidates at positions 1 and 3 are now eliminated. There is only one candidate remaining, on position 2. This is the new leader.

The board is extremely happy they now have a way to choose a new leader!

Little do the board know of Yang's wicked plans. He will calculate the position of the new

leader on the initial line and place himself there.

Input specification

The input consists of a single number N on a single line, the number of candidates showing up.

Output specification

Output on a single line, one number, i, Yang's position on the initial line.

Constraints

$1 \leq N \leq 10^{15}$

$1 \leq i \leq N$.

Sample input 1

```
5
```

Sample output 1

```
3
```

Sample input 2

```
1
```

Sample output 2

```
1
```

Sample input 3

```
1000000000000000
```

Sample output 3

```
874100093157377
```

Flaps on a plane

Maverick is a hotshot pilot, working his way up the ranks in an airline. Lately Maverick has had difficulties getting his landings right. When landing, Maverick likes to rely on the flaps for breaking. The position of the flaps p is given as a real number between 0 and 1, where 1 means fully retracted.

In an attempt to gain more control over his landings, Maverick has come up with the following model for the speed of the aircraft.

$$v(v_0, p; t) = v_0 p^{-p}(t + p)^p e^{-t}$$

This formula describes the speed of the aircraft at time t, given the landing velocity v_0 and the flaps position p. The formula assumes that the aircraft touches down at $t = 0$. Maverick doesn't know anything about formulas, but he wants to know how far the aircraft moved up the runway at time t, given v_0 and p. The distance the aircraft has moved since touch down is described by the following formula.

$$s(v_0, p; t) = \int_0^t v(v_0, p; t)\,\mathrm{d}t$$

Help Maverick out by writing a program that calculates how far the aircraft has moved since it touched down, given v_0, t and p.

Input

Input consists of a single line containing three real, positive numbers v_0, t and p, respectively.

Output

Output the position of the plane at time t if the flaps are in position p and the landing velocity is v_0.

Note: Output must have an absolute error less than 10^{-7}. So if s_0 is correct, then all answers s such that $|s - s_0| < 10^{-7}$ will be accepted.

Constraints

$0 \le v_0, t \le 100$

$0 < p \le 1$

Sample input 1

```
100 1 0.2
```

Sample output 1

```
77.79364224
```

Sample input 2

```
100 100 1
```

Sample output 2

```
200.00000000
```

Flaps on a plane 2

(see *Flaps on a plane* for full introduction)

It turns out Maverick isn't as good as he thinks, and needs more help finding the correct flap position. As before, Maverick is assuming that the speed of the aircraft is as described by the formula below.

$$v(v_0, p; t) = v_0 p^{-p} (t + p)^p e^{-t}$$

For each airport, there is an ideal position s (relative to where the aircraft touched down) for the aircraft to stop. The stopping position of the plane is given by the formula below, which depends on v_0 and p.

$$s(v_0, p) = \int_0^\infty v(v_0, p; t)\, \mathrm{d}t$$

Write a program that finds the correct flap position p so the aircraft stops in the ideal position. You can assume that such a position exists.

Input

Input consists of a single line containing two real, positive numbers v_0 and s.

Output

Output p, the position the flaps must be in for the aircraft to stop at position s, given v_0, the landing speed.

Note: Output must have an absolute error less than 10^{-7}. So if p_0 is correct, then all answers p such that $|p - p_0| < 10^{-7}$ will be accepted.

Constraints

$0 \leq v_0 \leq 100$

$v_0 \leq s \leq 2\, v_0$

$0 < p \leq 1$

Sample input 1

```
100 150
```

Sample output 1

```
0.32532542
```

Sample input 2

```
50 70
```

Sample output 2

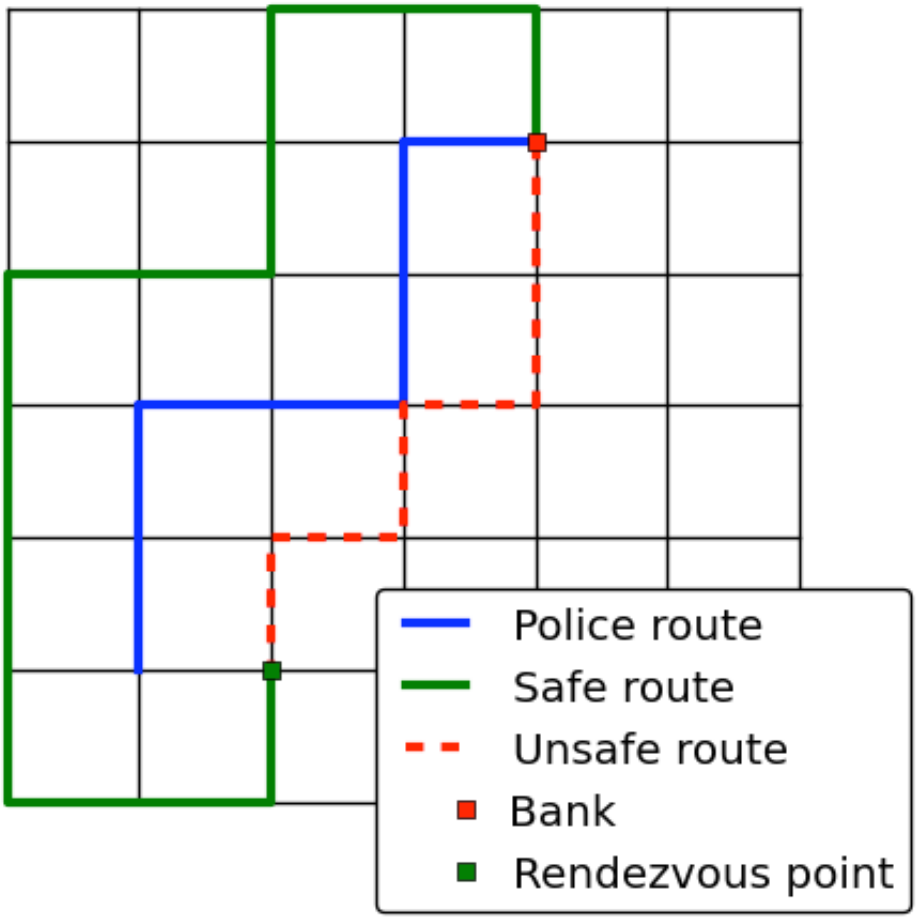
```
0.23039472
```

The getaway driver

Frank, a professional getaway driver, is hired to safely transport some robbers from a heist at the local bank. Getting away safely is quite easy, you just have to stay away from the police at all times. When the alarm goes off, the police will drive to the bank. Having a snitch within the police force, Frank knows exactly which route the police are going to take, so staying away from them should be easy.

Frank doesn't know when the police leave the police station (sometimes they are busy when the alarm goes off), nor how fast they are going. So the only way to stay entirely safe is to avoid their route all together. In practice this means that Frank cannot drive via any intersection that the police drive via (except the one at the bank, of course). This way he is sure he'll never get caught.

An example of this can be seen in the image below, where the green (and long) route is safe. The red route however, is unsafe because it goes via an intersection on the police route.



In order to put a fair price on the job, Frank needs to now how far he has to go in order to make a safe getaway. Make a program that outputs the length (in meters) of the shortest possible route Frank can take from the bank to the rendezvous point, while still being safe. If no such route exists Frank should back away from the deal.

The streets in city Frank lives in are layed out as a grid. It consists of n streets going west-east and m streets going north-south. The streets have an even spacing of `10m`, so the distance from an intersection to any of its neighboring intersection is always `10m`.

The bank is located at intersection $[b_i, b_j]$
The rendesvous point is at instersection $[r_i, r_j]$

Constraints

$2 < n, m \leq 100$
$0 < k \leq n \cdot m$
$0 \leq b_i, r_i < n$
$0 \leq b_j, r_j < m$

Input

The first line of input contains two integers n and m, the size of the city.
The second line contains four integers b_i, b_j, r_i and r_j. This is the location of the bank and the rendezvous point, respectively.
Then follows k lines describing the route that the police are taking. Each line contains two integers p^l_i and p^l_j, the coordinates of the l-th intersection on the route of the police.

Output

Output the shortest possible distance Frank has to drive from the bank to the rendezvous point without getting caught. If no such route exists, simply output `no deal`.

Sample input 1

```
7 7
1 4 5 2
5 1
4 1
3 1
3 2
3 3
2 3
1 3
1 4
```

Sample output 1

```
60
```

Sample input 2

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

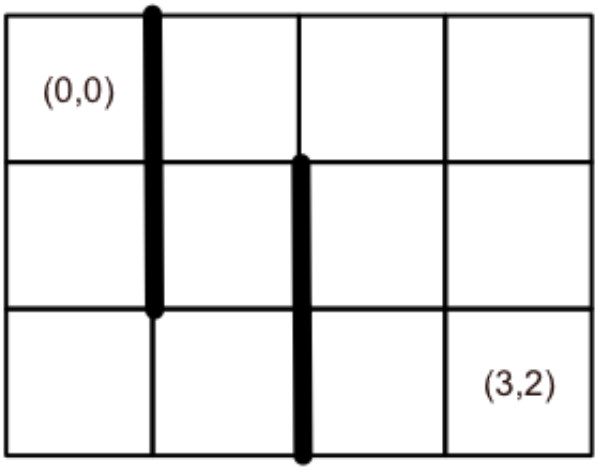
Sample output 2

```
no deal
```

Jail

So you find yourself trapped in a prison. However, together with your inmates, you have deviced a plan to escape the prison guards and win your freedom!

In order to safely escape to the pickup car waiting, which will bring you and your comrades to freedom, you must cross the prison yard. The prison yard is a rectangular grid of dimensions *M* times *N* divided into unit squares. You and your inmate friends are at the *(0,0)* square, and you want to get to the pickup car which will be at *(M-1, N-1)*. However, the prison guards will be chasing you. When escaping, you will not necessarily take optimal decisions. In fact, you know you will always increase exactly one your coordinates by *1* at each step, running towards the rendez-vous point. This makes things a bit difficult, but luckily, your inmates will sacrifice themselves for your freedom. Whenever you encounter a wall, on of your inmates will stay behind to help all the others climb over.



Now, given a map of the prison yard you wish to calculate how many of you can make it.

Input:

The first line of input contains 3 integers, *M*, *N* and *I*, denoting dimensions of the prison yard, and the number of inmates escaping (including you). The following *N* lines contains *M-1* numbers that are all 0 or 1. There is a prison wall between square (i,j) and (i+1,j) if and only if the i'th number on the j'th line is 1. Then follows *N-1* lines with *M* numbers on each line, telling that there is a prison wall between (i,j) and (i, j+1) if and only if the i'th number on the j'th line is 1. Square coordinates and lines are here 0-indexed.

Output:

Output one number on a single line, the maximal number of inmates that may reach the pickup point (including you).

Constraints

$2 \le M,N \le 300$

$1 \le I \le 10^9$

Sample input 1

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

Sample output 1

```
3
```

Sample input 2

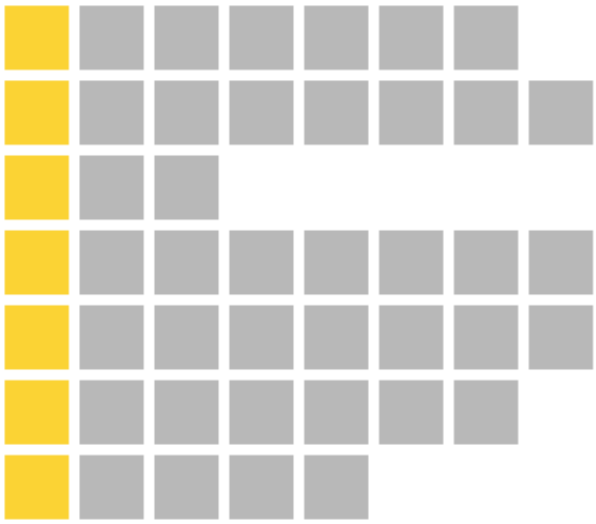
```
4 5 1
1 1 0
1 1 1
0 0 1
1 0 0
0 0 0
0 0 0 0
0 0 0 0
1 0 1 1
1 0 1 1
```

Sample output 2

```
1
```

Nuts

Johnny is working as the executive mathematician at NRK's *Påskenøtter* division. At NRK they have been struggling with having too many people solve all of the puzzles, and as a result they've asked Johnny to device new schemes for *Påskenøtter*.



Since Johnny isn't particularily creative, his main focus has been to investigate the number of ways to fill out the board. A board consists of r rows, each of which can have a different number of cells, c_i. Johnny has also been experimenting with using different alphabets, so each cell can hold one of n characters.

Johnny has now come up with a lot of different combinations of boards and alphabets, but has no idea which one to choose. Since the goal is to make *Påskenøtter* as difficult as possible, Johnny needs your help selecting the scheme which can be filled out in the largest number of ways. Of course, most of these "solutions" won't be actual solutions, they'll just be gibberish, but we don't care about that for now.

Input

Input consists of k lines.

Each line starts with an integer n, the number of characters in the alphabet. Then follows r integers c_i, which describes how many cells there are in row i.

Output

You should output the index (0-indexed, that is) of the scheme with the most number ways to fill out. You can assume that all schemes have a unique number of ways to fill out.

Constraints

$0 < k \leq 10^3$

$0 < r \leq 10$
$0 < n, c_i \leq 10^6$

Sample input

```
29 5 5 5
29 4 5 6 7
29 7 8 19 20 10
```

Sample output

```
2
```

QeuroPark [easy]

After getting fired from several large parking operator companies you've decided enough is enough; you'll start your own parking operator company - QeuroPark.

Even though you didn't exactly excel at your previous jobs, you know what it takes to be an efficient parking operator. There are three main factors affecting the success of such a company: A skilled and daring business crew, a motivated corps of inspectors and a super efficient call center, handling appeals and complaints.

Seeing as you are both daring and motivated, the only thing missing is an efficient call center. This is where your programming skills come into play. Emails received at the call center typically belong to one of two categories: *complaint* and *other*. If an email is categorized as *other*, then there's no way around taking a look at it yourself to decide if it's important. If the mail is categorized as a *complaint* on the other hand, it's easy; complaints are rejected.

Write a program that based on the subject line of an email, decides what action should be taken. You can assume that all complaint emails have `Complaint` as subject line.

Input

The first line of input contains a single integer n, the number of emails your program should process.

Then follows n lines, each representing the subject line of an email. The subject line is either `Complaint`, or some non-empty string.

Output

For each email output whether you should `Reject` it or `Investigate` it.

Constraints

$1 \leq n \leq 10^4$

Sample input

```
10
Complaint
Welcome to IAESTE's Career Fair
Ways to make your love more passionate
Complaint
Complaint
Contact Western union
You have received a postcard!
Complaint
Has Anyone Really Been Far Even
Complaint
```

Sample output

```
Reject
Investigate
Investigate
Reject
Reject
Investigate
Investigate
Reject
Investigate
Reject
```

Railway management [hard]

The young nation of Utopia have recently finished building their first railway, running all the way from the capital Aipotu to the city of Erewhon. The railway is split into *n* segments, numbered from `0` to `n - 1`, where Aipotu is located at segment `0` and Erewhon is located at segment `n - 1`. Along the railway there are many small cities and villages, each connected to some railway segment. All of the trains depart either from Aipotu or Erewhon, since all the other cities along the railway are too small to have their own train yard.

Since the utopians are quite inexperienced in the art of operating a railway, there tends to be quite a few problems with the rails, or broken trains blocking the railway and so on. Therefore they need you to build an information system that can tell them whether a particular rail segment currently is reachable from either terminal station.

The system will receive three types of messages:

- Error report
  stating that a specific rail segment is blocked
- Repair report
  stating that a specific rail segment is reopened after being blocked
- Status request
  querying whether a specific rail segment is reachable from Aipotu or Erewhon

The system should take note of all the reports, and output either `good service` or `no service` to each status request. A segment has `no service` if the segment itself is blocked, or there is a blockage between the segment and both of the terminal stations.

At the beginning of the day all of the segments of the railway line are assumed to be opened.

Input specification

The first line of input contains a single integer *n*, the number of segments on the railway line.

Then follows *k* lines, representing the messages. Each line contains an integer *i*, the index of

the segment the message is referring to, and a word *w*, the type of message. *w* will be either `blocked`, `opened` or `status`. There will only be issued `blocked` messages for segments that are already opened, and visa versa.

Output specification

For each line containing a `status` message your program should output either `good service` or `no service`, according to the railway's current status.

Constraints

$2 \leq n \leq 10^5$
$0 \leq k \leq 10^5$
$0 \leq i < n$

Sample input

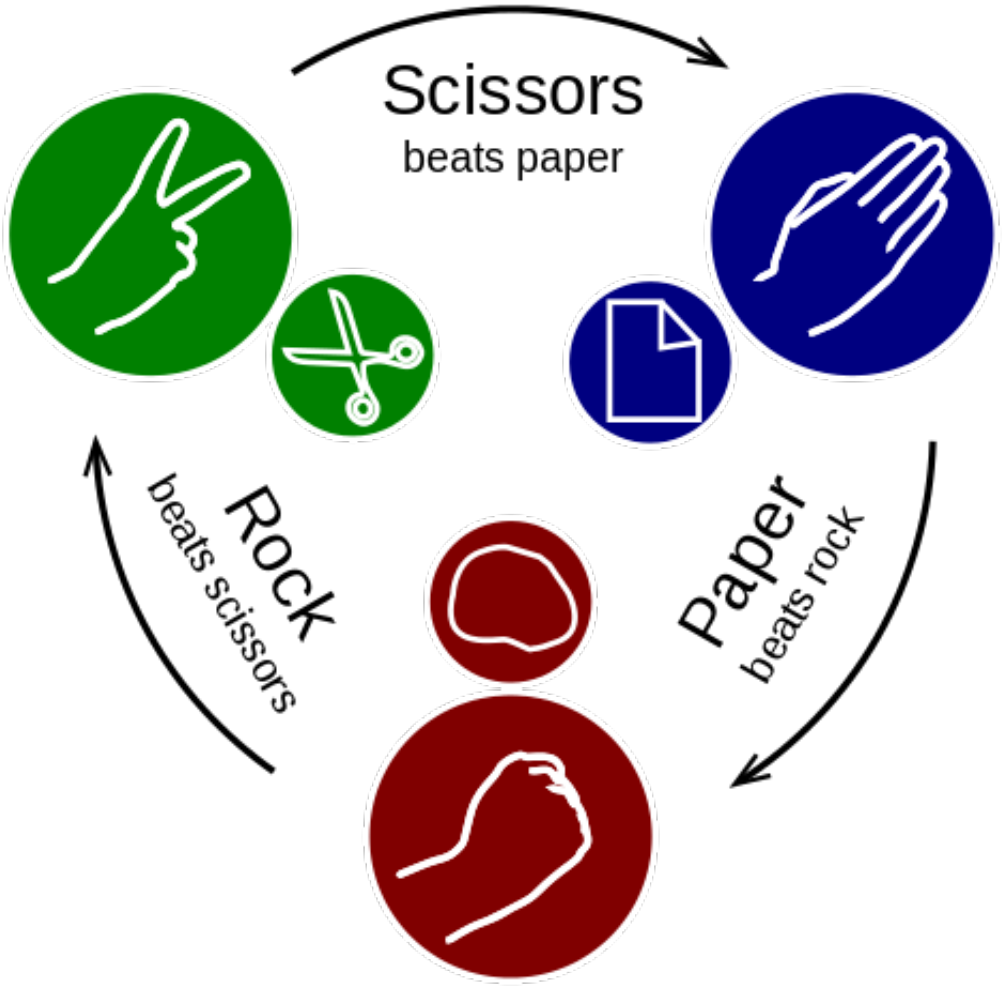
```
5
0 status
2 status
4 status
2 blocked
2 status
3 status
4 blocked
3 status
4 opened
3 status
```

Sample output

```
good service
good service
good service
no service
good service
no service
good service
```

Rock, paper, scissors [hard]

The old game of Rock-Paper-Scissors goes as follows: Each of two players chooses either "Rock", "Paper", "Scissors" and they expose their choice simultaneously. Then if they have chosen the same sign, the game is drawn and they will have to play again. If not, the rules are that rock beats scissors, scissors beat paper and paper beats rock.



Your friend just came up with a new version of the game and it goes as follows: You play three rounds of Rock-Paper-Scissors, where you have to use every sign exactly once during the three rounds. The winner of the game is the winner of the third round.

Now your friend wants to try this new game with you and he wants to play for 10000 rounds. For each game, you will get a score of 1 point if you win, 0.5 points if you draw and 0 points if you lose. You don't want to make a fool of yourself, so you want to score more than 4750 points in (almost) any case. In other words, you want to beat your friend for each possible strategy he may choose. Your task is to write a program for achieving this.

Input

The first line of input will contain an integer *T*, the number of test cases. Then for each test case, there will be 10000 lines with two space separated words drawn both with uniform probability from the set {"Rock", "Paper", "Scissors"}, the first play of a round from you and your friend, respectively.

Output

For each test case, output 10000 lines, each line containing a single word. The word on the i'th line denoting your second play on the i'th round. (Your third play will then be completely determined).

The output for a test case will be considered correct if all your plays are valid, and if your total score is larger than 4750. Your solution will be considered correct if it gives a correct output to each test case.

Constraints

$1 \leq T \leq 100$

Sample input

```
1
Rock Paper
Paper Paper
Rock Scissors
Scissors Rock
...+9996 more lines
```

Sample output

```
Paper
Rock
Paper
Rock
...+9996 more lines
```

Sample opponent play (Hidden)

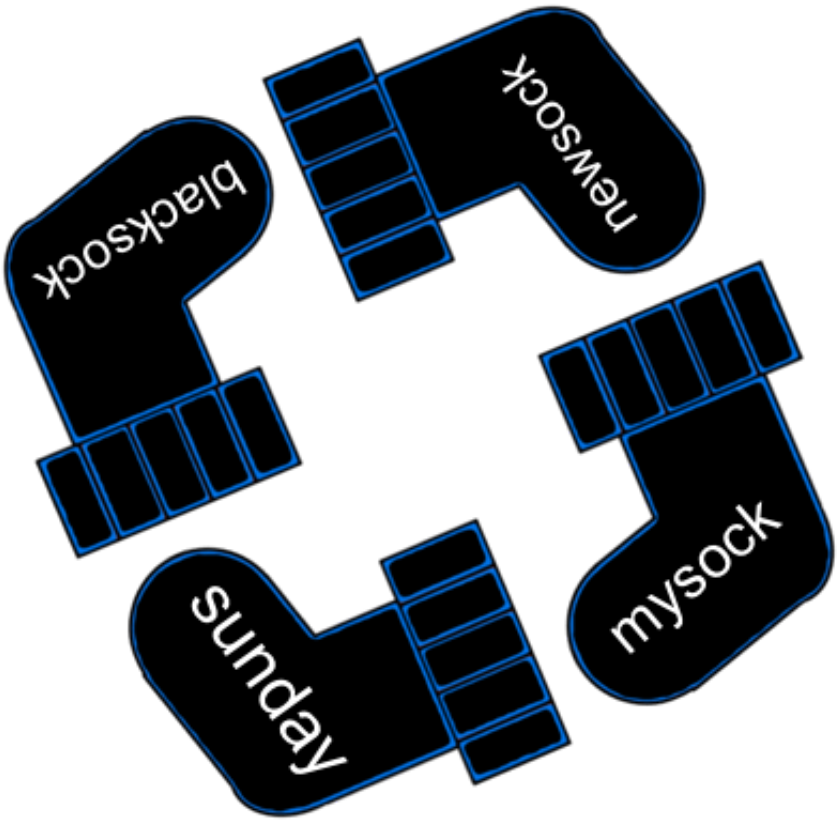
```
Scissors
Rock
Rock
Paper
...+9996 more lines
```

Sample score

```
0-1 (Scissors-Rock)
1/2-1/2 (Scissors-Scissors)
1-0 (Scissors-Paper)
0-1 (Paper-Scissors)
...+9996 more lines
```

Socks

Matias works as a programmer, and like most programmers these days, he spends a lot of time working in his terminal window. When Matias is not at work, he spend his time figuring out what to wear at work the next day, because he thinks it is essential to be classy and good looking at work. Since the background of his terminal is black, he always wear black socks at work to make sure that he matches the computer.



With a drawer full of black socks, it is very hard to find matching pairs, so Matias has come up with a very smart system. Every time he buys a new pair of socks, he picks a random word and make sure that both of the new socks are labeled with that word. This way, he can easily check if two socks are a match, simply by looking at the word written on the socks. Once in a while he buys many socks that are all equal. Then it does not really matter which of them he pairs together, so he labels them all with the same word.

Like every other person, Matias has a washing machine which eats socks. So every time he has washed them, he collects them all to throw away socks that can not be paired together with any other sock. Matias wants to make a program to do this boring task for him, but since he is not very good with algorithms (he only configures servers at work), he asks you to help him. Help Matias to throw away as few socks as possible while making sure that each remaining sock can be paired with at least one other sock.

Input:

The first line of input contains one integer *N*, the number of socks Matias has washed today.

Then follows *N* lines, each containing one string. The string on line *i* represents the text written on the *i*'th sock. The strings will only contain the lowercase letters from the english alphabet.

Output:

Output a list sorted alphabetically with all the socks Matias should remove. If he does not need to remove any socks, output "Sock-sess"

Constraints

$1 \leq N \leq 100$

Sample input 1:

```
6
tuesday
monday
tuesday
monday
saturday
saturday
```

Sample output 1:

```
Sock-sess
```

Sample input 2:

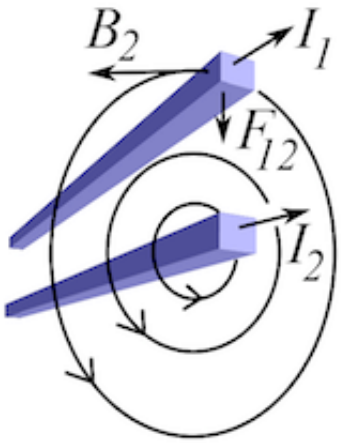
```
4
bluesock
redsock
bluesock
bluesock
```

Sample output 2:

```
redsock
```

Wired

Jan, an electrician, likes to design electrical circuits. As you know, around any conductor with a current, a magnetic field is induced. Jan must take this into account when he designs his electrical circuits, otherwise they might break due to forces between the wires.



Jan has decided to use duck tape to strap the wires to the circuit board. Unfortunately, the duck tape Jan can only withstand lateral forces (forces that are orthagonal to the wire, that is). Taking all the physics into account, this means that the circuit will break unless all pairs of wires are either perfectly parallell or orthogonal.

Write a program that decides if the circuit will break or not, given one of Jan's designs. There is no room for error, so make sure that your calculations are accurate (i.e. don't use any floats!).

Input

The first line of input contains a single integer n, the number of wires.

Then follows n lines, each describing a wire. Each such line contains four integers x_1, y_1, x_2, y_2, representing a wire which goes through the points (x_1, y_1) and (x_2, y_2) on the circuit board. (x_1, y_1) and (x_2, y_2) will not be equal, and no wires will be duplicate.

All wires extend to the end of the circuit board, so for all practical concerns they might be considered to be infinitely long.

Output

If the circuit will break, output `broken`, otherwise output `wire it up`.

Constraints

$1 \leq n \leq 10^4$
$|x_1|, |x_2|, |y_1|, |y_2| \leq 10^5$

Sample input 1

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

Sample output 1

```
wire it up
```

Sample input 2

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

Sample output 2

```
broken
```

Saying hello [sample]

You want to test your programming skills by making a modified version of the classic `Hello, world!`-program. Your program should read a number *n*, and then print *n* lines containing `Hello, world!`.

Input

Input consists of a single line with an integer *n*, the number of `Hello, world!` you should output.

Output

Output *n* lines containing `Hello, world!`.

Constraints

$1 \leq n \leq 10^3$

Sample input 1

```
1
```

Sample output 1

```
Hello, world!
```

Sample input 2

```
5
```

Sample output 2

```
Hello, world!
Hello, world!
Hello, world!
Hello, world!
Hello, world!
```

Solution

```
from sys import stdin

n = int(stdin.read())

for i in xrange(n):
    print 'Hello, world!'
```

Summing up [sample]

You have a lot a text containing ten integers on each line, and you really want to calculate the sum of these ten integers for each line.

Make a program that calculates the sum of the integers on each line.

Input

Input consists of n of lines, each containing 10 integers i_j.

Output

For each line of input, output the sum of all i_j on that line.

Constraints

$1 \leq n \leq 10^3$
$0 \leq |i_j| \leq 10^6$

Sample input 1

```
1 1 1 1 1 1 1 1 1 1
```

Sample output 1

```
10
```

Sample input 2

```
1 1 1 1 1 1 1 1 1 1
2 2 2 2 2 2 2 2 2 2
0 1 2 3 4 5 6 7 8 9
-5 -4 -3 -2 -1 1 2 3 4 5
```

Sample output 2

```
10
20
45
0
```

Solution

```
from sys import stdin

# loop through all the input lines
for line in stdin:
    numbers = map(int, line.split()) # parse all integers on current line

    print sum(numbers)
```