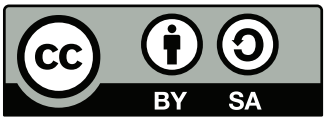
BAPC 2020 Preliminaries

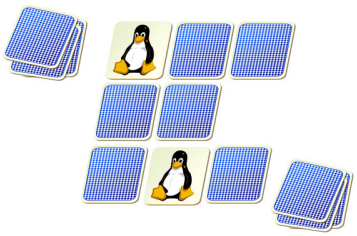
Preliminaries for the
2020 Benelux Algorithm Programming Contest

Problems

A	Adversarial Memory
B	Binary Seating
C	Cutting Corners
D	Ducky Debugging
E	Eightgon
F	Figure Skating
G	Group Project
H	Human Pyramid
I	In-place Sorting
J	Jam-packed
K	Kangaroo Commotion



A Adversarial Memory



Charlie is playing a game of *memory* (also known as *concentration*) on his own. The game consists of $2n$ cards, where each of the numbers from 1 to n is written on exactly two cards. The cards are upside down on the table. In a turn, Charlie turns over one card, looks at it, and then turns over another card. If the cards have the same number, they are removed from the game. Otherwise, they are turned back over and placed back. The goal of the game is to remove all the cards from the game in as few moves as possible.

You are a magician, and hence you are able to change the numbers on upside down cards seamlessly. If Charlie turns over the same card twice, you need to make sure that he sees the same number both times, or else Charlie would notice something is wrong. You also need to make sure that for each number there will be exactly two cards on which Charlie will see that number. Your goal is to force Charlie to need at least $2n - 1$ turns to finish the game.

More formally, there are $2n$ indices from 1 to $2n$. In a turn, Charlie chooses an index i and turns over the card at index i. You can then decide what number Charlie will see when he turns over the card. Then Charlie will choose a different index j and turn over the card at index j. You can then decide what number Charlie will see when he turns over that card. The only restrictions are that Charlie must always see the same number when he turns over the same card, and that for each number there will be exactly two cards on which Charlie will see that number. Note that Charlie will never choose the index of a card that is already out of the game.

Interaction

This is an interactive problem. Your submission will be run against an *interactor*, which reads the standard output of your submission and writes to the standard input of your submission. This interaction needs to follow a specific protocol:

The interactor first sends a line containing the integer n ($1 \leq n \leq 10\,000$), the number of different values on the cards.

Following this, $2n - 1$ turns will be played. Each turn consists of two parts:

- The interactor first sends a line containing an integer i ($1 \leq i \leq 2n$), denoting the index of the first card Charlie turns over. Your program must respond with the number on card i.
- The interactor then sends a line containing an integer j ($1 \leq j \leq 2n$), denoting the index of the second card Charlie turns over. Your program must respond with the number on card j.

For each turn, it is guaranteed that i and j are distinct indices of cards that are still in the game.

Your submission should exit after it has answered $2n-1$ turns (i.e. printed $4n-2$ numbers). Reading more input will result in a time limit exceeded and printing more output will result in a wrong answer.

Make sure you flush the buffer after each write.

A testing tool is provided to help you develop your solution.

Read	**Sample Interaction 1**	**Write**
1 1		
	1	
2		
	1	

Read	**Sample Interaction 2**	**Write**
3 1		
	1	
2		
	2	
3		
	2	
2		
	2	
4		
	3	
5		
	1	
5		
	1	
1		
	1	
4		
	3	
6		
	3	

B Binary Seating

By accident, two rooms (room `0` and room `1`) got booked for the theoretical exam of the B++ Applied Programming Course and both were communicated to the students. Now students might go to either of the rooms, and as a student assistant your job is to supervise room `1`. Since you assisted all these students during the course, you know how much time each student will need to finish the exam. Already before the exam you are eager to go home, but you can only leave when all of the students in your examination room have finished. You assume that every student chooses one of the exam rooms with equal probability, independent of the other students. After how much time do you expect to be able to leave?

By Freepik on www.flaticon.com

Input

The input consists of:

- A line with an integer n ($1 \leq n \leq 40$), the number of students.
- A line with n integers $t_1, \ldots, t_n$ ($1 \leq t_i \leq 1000$): t_i is the time it takes for the ith student to finish the exam and leave.

Output

Output the expected time before you can leave. Your answer should have an absolute or relative error of at most 10^{-6}.

Sample Input 1	**Sample Output 1**
`2` `2 3`	`2`

Sample Input 2	**Sample Output 2**
`5` `1 4 5 2 3`	`4.03125`

Sample Input 3	**Sample Output 3**
`5` `2 1 1 1 1`	`1.46875`

C Cutting Corners

A large coffee spill in the warehouse of the Busy Association of Papercutters on Caffeine has stained the corners of all paper in storage. In order to not waste money, it was decided that these dirty corners should be cut off of all pieces of paper.

CC BY-SA 2.0 by SixRevisions on Flickr

A few members loudly proclaim that cuts should be made diagonally, while other members say that cutting the corner out as a rectangle is the better option. Both parties claim their method is better.

You decide to end this discussion once and for all, telling the rectangle-cutters that their method is slower. You set out to show them the following: given a piece of paper which has a w by h corner that is stained with coffee that needs to be to cut off, how much more effort is it to cut out the whole rectangle compared to cutting along the diagonal?

Input

The input consists of:

- A line containing two integers w and h ($1 \leq w, h \leq 100$), representing the width and height of the corner respectively.

Output

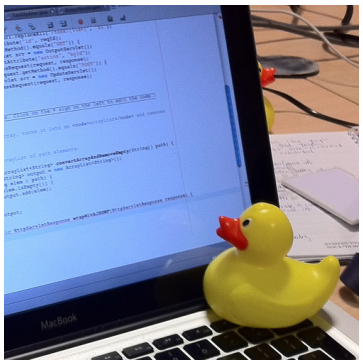
Output how much longer you have to cut if you cut out a rectangle, compared to cutting along the diagonal. Your answer should have an absolute or relative error of at most 10^{-6}.

Sample Input 1	**Sample Output 1**
`3 4`	`2`

Sample Input 2	**Sample Output 2**
`12 7`	`5.107556011`

Sample Input 3	**Sample Output 3**
`1 1`	`0.585786438`

D Ducky Debugging


CC BY-SA 3.0 By Tom Morris on Wikipedia

You don't know anything about programming, but nonetheless you and your friend Bob joined the BAPC. Bob just submitted a solution to a problem, but sadly his submission was not accepted. You will help him figure out what mistake he made. In order to do so, he will explain what every part of the program should do according to him. He does not want you to interrupt him while he is explaining, but whenever he is unsure of something, he will ask for confirmation. Bob is actually a very good coder, so whenever he asks for confirmation, he is simply right. You just need to react by nodding and saying "Quack!". As long as you don't interrupt him at the wrong time, and as long as you say "Quack!" at the right times, he will eventually find his mistake and he will shout "I quacked the code!".

Interaction

This is an interactive problem. Your submission will be run against an *interactor*, which reads the standard output of your submission and writes to the standard input of your submission. This interaction needs to follow a specific protocol:

The interactor sends one line of text at a time, each line being one sentence of Bob's monologue. You need to reply as follows:

- If the line ends with a question mark ("`?`"), your reply must be the string "Quack!".
- If the line ends with a period ("`.`"), your reply must be the string "*Nod*".
- If the line is exactly "I quacked the code!", your submission should exit.
  Reading more input will result in a time limit exceeded and printing more output will result in a wrong answer.

Bob's monologue will be no more than 100 sentences, with at most 10 000 characters in total. It is guaranteed that every sentence, except the last one, ends with a question mark ("`?`") or a period ("`.`").

Make sure you flush the buffer after each write.

A testing tool is provided to help you develop your solution.

Read	**Sample Interaction 1**	**Write**
`In the loop, I read a line of input.`		
		`*Nod*`
`The method std::getline(std::cin, line); reads until a newline, right?`		

```
                                                    Quack!
Then I branch according to the last
character of the line of input.
                                                    *Nod*
When the character is an exclamation
mark, I terminate the program.
                                                    *Nod*
When it is a question mark, I need
to print "Quack!" right?
                                                    Quack!
In any other case, I simply print
"*Nod*".
                                                    *Nod*
Wait, to flush the output stream
I need to use std::endl instead of
'\n' right?
                                                    Quack!
I quacked the code!
```

Read **Sample Interaction 2** **Write**

```
My program seems to always exit
after reading one line.
                                                    *Nod*
Shouldn't it only halt when a line
ends with an exclamation mark?
                                                    Quack!
After reading a line, I check the
last character using a switch.
                                                    *Nod*
When the character is a period, I
print "*Nod*".
                                                    *Nod*
When the character is a question
mark, I print "Quack!".
                                                    *Nod*
Wait, both get printed before the
program exits?
                                                    Quack!
I guess I forgot to add break; here
and there...
                                                    *Nod*
I quacked the code!
```

E Eightgon

After many years you and your coauthor H. Addaway have finally developed a Theory of Everything that explains everything: Why does time have a direction? How should quantum mechanics be interpreted? What caused the Big Bang? What is love?

An unfortunate fact about physics is that physical theories need to be experimentally tested. In particular, your theory rests on the discovery of so called Barely Audible Particle Clusters (BAPCs). For this purpose you have proposed the development of a Large Eightgon Collider. What remains is to find a suitable location to construct this scientific wonder.

For obvious reasons, the Large Eightgon Collider must consist of eight straight tunnels that together form an underground cycle. Each tunnel is allowed to have a different non-zero length. At each of the eight tunnel connections, a special detector must be built, that also slightly deflects the particles 45 degrees to the left. Each of the eight detectors attracts many researchers, requiring a shaft to the surface to supply them with fresh food and oxygen.

In order to save costs, they will reuse abandoned mine shafts. Given a map of all abandoned mine shafts, your job is to find the number of possible locations to build this miracle. You only consider locations where at least one tunnel runs parallel to the x-axis of the map.

Figure E.1 shows the second sample.

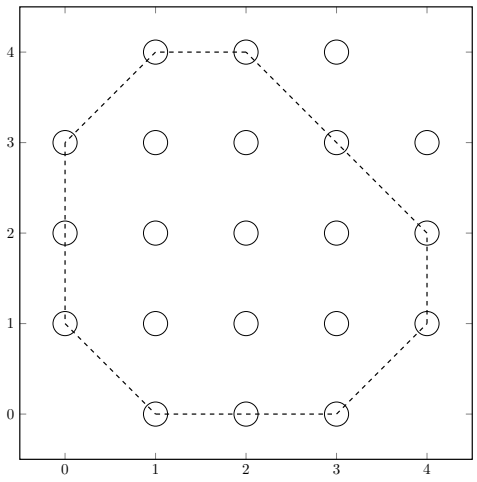


Figure E.1: Visualisation of Sample 2 showing one possible location for the Large Eightgon Collider.

Input

The input consists of:

- A line with an integer n ($1 \leq n \leq 5000$), the number of abandoned mine shafts.
- n lines, each with two integers x and y ($-10^8 \leq x, y \leq 10^8$), the coordinates of the abandoned mine shafts.

Output

Output the number of possible locations to build the Large Eightgon Collider.

Sample Input 1	Sample Output 1
8 0 1 1 0 0 2 2 0 3 1 1 3 3 2 2 3	1

Sample Input 2	Sample Output 2
21 0 1 0 2 0 3 1 0 1 1 1 2 1 3 1 4 2 0 2 1 2 2 2 3 2 4 3 0 3 1 3 2 3 3 3 4 4 1 4 2 4 3	15

F Figure Skating

CC-BY-SA 4.0 By Sandro Halank on commons.wikimedia.org

Figure skating is a very popular sport at the Winter Olympics. It has been on the programme the longest of all winter sports, having even been included in the Summer Olympics before the split in 1924. Just like in gymnastics, each contestant executes a routine consisting of elements, which are individually scored by a jury. This subjective aspect to judging skill always leaves room for heated discussion, but a huge scandal in the 2002 Winter Olympics, with allegations that the game had been fixed, caused a transition to the new scoring system IJS. Points awarded to each element of the routine are known beforehand: A Lutz scores 0.60 points (but 2.10 for a double and 5.90 for a triple), a Salchow scores 0.40 (1.30 for double, 4.30 for triple), an Euler scores 0.50, et cetera. Then, points are added or subtracted by the jury based on execution. Consequently, a figure skater is able to estimate his or her score assuming average performance.

Olympics observers from the Bookmakers' Association for the Prevention of Cheating are tasked with assessing the objectivity of the jury. They will compare the predicted ranking of the contestants with the final outcome to determine who is the jury's favourite. The favourite is the contestant who rose the most places between the predicted and final scoreboard. Ties are broken by whoever ends up higher on the final scoreboard. However, if no one did better than predicted, this raises some red flags with the observers, which is declared "suspicious".

Input

The input consists of:

- A line containing a single integer n ($1 \leq n \leq 1000$), the number of contestants.
- n lines, the ith of which contains the name of the contestant who places ith on the predicted scoreboard.
- n lines, the ith of which contains the name of the contestant who places ith on the final scoreboard.

Each name consists of at most 100 lower-case and upper-case alphabetical characters. All names are unique, and occur on both scoreboards exactly once.

Output

If the scoreboards are suspicious, output "`suspicious`". Otherwise, output the name of the jury's favourite.

Sample Input 1

```
3
Plisetsky
Katsuki
Leroy
Leroy
Plisetsky
Katsuki
```

Sample Output 1

```
Leroy
```

Sample Input 2

```
2
Allison
Bobson
Allison
Bobson
```

Sample Output 2

```
suspicious
```

Sample Input 3

```
3
daSilva
Aziz
Peters
Aziz
Peters
daSilva
```

Sample Output 3

```
Aziz
```

G Group Project

CC-BY-NC 2.0 By Mark Knobil, knobil on Flickr

The big day has finally arrived: today you are going to form groups of two in which you will do the end-of-the-year project. When you arrive at school, you learn that the teacher of the other class is sick, and that your teacher, Mr. B.A.P. Cee, will also have to make groups for the other class. Mr. B.A.P. Cee is a smart guy and realizes that he can use these unfortunate circumstances to his advantage.

Ending up with groups of one should be avoided at all cost, so mixing the students of the two classes may avoid this situation. However, while it is easy to pair up two students from the same class, it is more difficult to match up students from different classes. Throughout the years there has been a lot of rivalry between the two groups, and many students dislike students in the other class. Mr. B.A.P. Cee knows which pairs of students will result in a fight and a failed project.

You are given a list of pairs of students who cannot work together. How many disjoint groups of two can Mr. B.A.P. Cee make that will not result in a failed project?

Input

The input consists of:

- A line with two integers n ($1 \leq n \leq 10^5$), the number of students, and m ($0 \leq m \leq 2 \cdot 10^5$), the number of pairs of students who cannot work together.
- m lines, each with two distinct integers i and j ($1 \leq i, j \leq n$, $i \neq j$), giving a pair of students who cannot work together.

Students are identified by the numbers 1 through n. It is guaranteed that it is possible to split the students into two classes in such a way that all students from the same class get along.

Output

Output the number of pairs of students Mr. B.A.P. Cee can make without making any pair of students who cannot work together.

Sample Input 1	Sample Output 1
3 2 1 2 3 1	1

Sample Input 2

```
5 6
1 4
2 4
3 4
1 5
2 5
3 5
```

Sample Output 2

```
2
```

Sample Input 3

```
6 6
1 4
2 5
3 6
1 5
3 5
2 6
```

Sample Output 3

```
3
```

H Human Pyramid

The Barefooted Acrobatics People's Club wants to make a group photo in an original way. For the photo, they want to make a human pyramid, where each person rests on the ground or rests on the shoulders of two people below him or her.

CC BY-SA 3.0 by Xzenia Witehira on Wikipedia

Making a human pyramid demands a lot from the acrobats involved, so the club selected a group consisting of strong people of which they are assured that these people can carry enough weight. The others are 'agile' and to make sure everyone is comfortable during the photo, there can only be agile people directly above an agile person.

The photographer wants to make a photo of a pyramid with h people on the floor, $h-1$ on the second layer, $h-2$ on the third layer, and so on, with a single person on the hth layer. You have s strong people at your disposal, and the other $\frac{1}{2}h(h+1) - s$ people are agile. What is the number of ways you can arrange the pyramid satisfying the demands of the photographer? Since this number may be large, you should find it modulo $10^9 + 7$.

Two pyramids P_1 and P_2 are different if there exists a location where P_1 has an agile person and P_2 a strong person, or vice versa.

Input

The input consists of:

- A line containing two integers h ($1 \leq h \leq 100$) and s ($0 \leq s \leq \frac{1}{2}h(h+1)$), the number of layers in the pyramid and the number of strong people.

Output

Output the number of possible ways to build a pyramid with the given constraints, modulo $10^9 + 7$.

Sample Input 1	Sample Output 1
3 3	3

Sample Input 2	Sample Output 2
5 3	14

I In-place Sorting

Woe is you – for your algorithms class you have to write a sorting algorithm, but you missed the relevant lecture! The subject was in-place sorting algorithms, which you deduce must be algorithms that leave each input number in its place and yet somehow also sort the sequence.

Of course you cannot change any of the numbers either, then the result would just be a different sequence. But then it hits you: if you flip a 6 upside-down, it becomes a 9, and vice versa! Certainly no one can complain about this since you changed none of the digits! The deadline to hand in the exercise is in five hours. Try to implement this sorting algorithm before then!

Input

The input consists of:

- A line with an integer n ($2 \leq n \leq 10\,000$), the number of integers in the input sequence.
- n lines, the ith of which contains a positive integer x_i ($1 \leq x_i \leq 10^{18}$), the ith number of the sequence.

Output

If the sequence cannot be sorted in non-decreasing order by flipping some of the digits 6 or 9 in the input[1], output "`impossible`". Otherwise, output "`possible`" followed by the sorted sequence – each number on its own line.

If there are multiple valid solutions, you may output any one of them.

Sample Input 1	Sample Output 1
4 9 7 7 9	possible 6 7 7 9

Sample Input 2	Sample Output 2
4 97 96 66 160	possible 67 69 99 190

[1] Flipping any of the digits of n is not allowed.

Sample Input 3

```
3
80
97
79
```

Sample Output 3

```
impossible
```

Sample Input 4

```
2
197
166
```

Sample Output 4

```
possible
167
169
```

J Jam-packed

CC-BY-SA 2.0 By treehouse1977 on Flickr

The fruit harvest has been good this year, which means that your jam-selling company, which produces the price-winning Berry Artisanal and Pure Compote, is shipping out jam left and right! A customer has recently placed a huge order of n jars of jam. To ship these jars, you put them into boxes, each of which can hold up to k jars.

As is always the case with fragile goods, the jars might break in the process of being delivered. You want to avoid the jars bouncing around in their boxes too much, as that significantly increases the chance that they break. To circumvent this, you want to avoid having boxes that are too empty: that would inevitably result in the uncontrolled bouncing around, and subsequently breaking, of the jars. In particular, you want the box with the least number of jars to be as full as possible. In order to estimate the risk you are taking with your precious jars, you would like to know: how many jars does this box contain?

Input

The input consists of:

- A line with two integers n ($1 \leq n \leq 10^{18}$), the number of jars that need to be packed, and k ($1 \leq k \leq 10^{18}$), the number of jars a single box can hold.

Output

Output the number of jars that the least filled box contains.

Sample Input 1	Sample Output 1
10 3	2

Sample Input 2	Sample Output 2
16 4	4

Sample Input 3	Sample Output 3
1 2	1

K Kangaroo Commotion

Bushfires are threatening your habitat! Being a kangaroo, you must inform the other kangaroos in your troop as fast as possible and flee to a safe area.

You are currently standing still, and you will jump to the other kangaroos' locations. You will visit the other kangaroos in a specific order so that all of them have sufficient time to escape. After visiting all other kangaroos you must continue jumping to the safe area, where you should come to a stop.

With each jump you move a (possibly negative) integer distance north and/or east. Because of your limited muscle power, you are only able to accelerate or decelerate at most 1 in each direction each jump. Formally, if jump i moves you $v_{x,i}$ to the north and $v_{y,i}$ to the east, the next jump $i+1$ must satisfy $|v_{x,i+1} - v_{x,i}| \leq 1$ and $|v_{y,i+1} - v_{y,i}| \leq 1$.

Find the minimal number of jumps needed to go from your current position to the safe area via all other kangaroos, without leaving the grid. It is very important to come to a full stop at the end, so the last jump must both start and end at the safe area.

The first sample is shown in figure K.1.

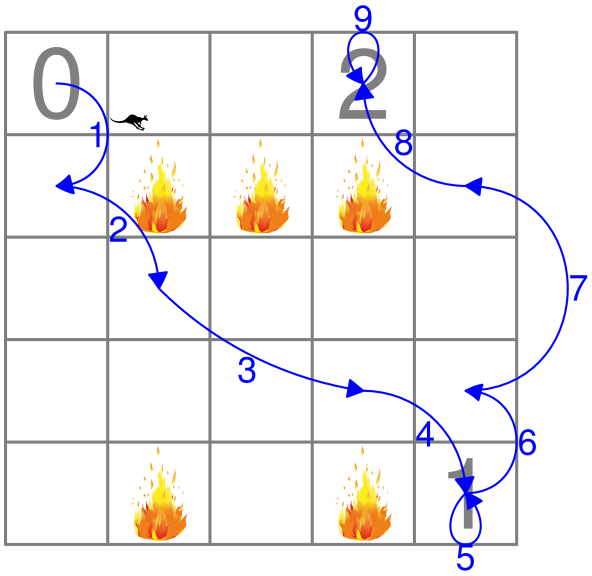


Figure K.1: Visualisation of Sample 1 showing one possible way to get to the safe area using 9 jumps.

Input

The input consists of:

- A line containing three integers r, c ($1 \leq r, c \leq 50$), the number of rows and columns of the grid, and k ($1 \leq k \leq 5$), the number of other kangaroos you need to warn.
- r lines each consisting of c characters. A "`.`" indicates an open space and a "`#`" indicates a bush where you can't jump.

Your start position is indicated by a "`0`" and the characters "1" to k indicate the positions of the other kangaroos you need to warn *in this order*.

The position indicated by $k+1$ indicates the safe area where you should come to a stop.

Output

If it is possible to reach the safe area, then output the minimal number of steps needed to reach the safe area. Otherwise, output "`impossible`".

Sample Input 1

```
5 5 1
0..2.
.###.
.....
.....
.#.#1
```

Sample Output 1

```
9
```

Sample Input 2

```
2 2 2
03
12
```

Sample Output 2

```
4
```

Sample Input 3

```
1 5 1
.0#21
```

Sample Output 3

```
8
```

Sample Input 4

```
3 4 1
#0##
#.#2
1###
```

Sample Output 4

```
impossible
```