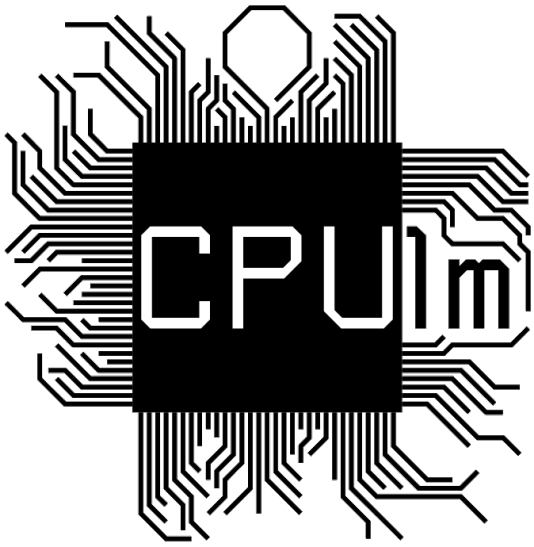
Winter Contest 2025

January 25th



Problems

Jan 25	Winter Contest *(easy)*
Jan 29	Chinese New Year
Mar 3	Carnival
Mar 14	Pi Day
Mar 21	Colour Day
Apr 20	Easter
Jul 5	Dependence Day
Jul 25	Sysadmin Day
Jul 30	Friendship Day
Nov 14	Domino Day
Nov 27	Thanksgiving *(easy)*
Dec 1	Advent
Dec 31	New Year's Eve *(very easy)*

This page is intentionally left (almost) blank.

Problem Jan 25: Winter Contest

Time limit: 2 seconds

Did you know that today is a very special day? No, not because of Winter Contest 2025, but because today is a *square day*, meaning that the year is a square number ($2025 = 45^2$), the month is a square number ($1 = 1^2$) and the day is a square number as well ($25 = 5^2$).

Year	Winter Contest	GCPC
2019	Jan 26	Jul 6
2020	Jan 25	Nov 21
2021	*N/A*	Jun 26
2022	Jan 29	Jun 25
2023	Jan 28	Jun 17
2024	Jan 27	Jun 22
2025	Jan 25	*TBA*

Figure Jan 25.1: Dates of Winter Contests and GCPCs in the past few years. This year's Winter Contest is the only one on a square day.

Such square days are generally quite uncommon. In fact, this year is the first year in which it is possible to host Winter Contest on a square day, ever since the very first programming contests dating back to 1970 (and Winter Contest only dates back to 2005 anyway). To find out just how uncommon these special days are, given a range of years, compute the total number of square days that fall within that range.

Input

The input consists of:

- One line with two integers a and b ($1970 \leq a \leq b \leq 9999$), where a is the first year of the range and b is the last year.

Output

Output the number of square days for which the year is between a and b inclusive.

Sample Input 1	Sample Output 1
`2025 2025`	`15`

Sample Input 2	Sample Output 2
`2026 2115`	`0`

Sample Input 3	Sample Output 3
`1970 6400`	`540`

Sample Input 4	Sample Output 4
`5555 9999`	`375`

This page is intentionally left (almost) blank.

Problem Jan 29: Chinese New Year

Time limit: 4 seconds

Image by DGJ, pixabay.com

Chinese New Year is in four days and Lina is already excited about all the festivities. She looks forward to eating dumplings on New Year's Eve and setting off fireworks to greet the new year. And of course, the money she gets in small red bags from her relatives is always a plus.

Her whole family is already busy with the preparations such as cleaning and decorating the house. Each year, they put up Chinese Spring Couplets next to their front door which will bring luck and prosperity next year. For the first time, it is Lina's job to write the couplets, and she takes it very seriously. However, Lina did not yet start her preparations and now she has to hurry up to finish before New Year's Eve. Instead of following the very strict rules in which order the strokes of a glyph have to be written, Lina decides to speed things up by bending the rules a little. She just wants to draw all the glyphs with as few strokes as possible without her parents noticing it.

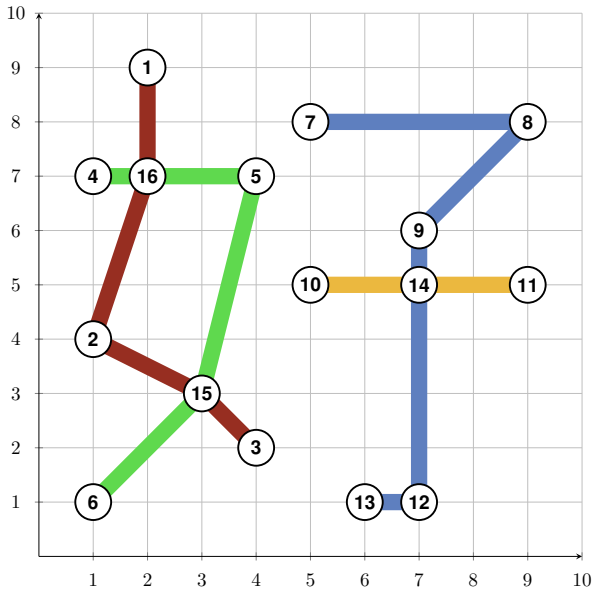


Figure Jan 29.1: Visualization of the second sample. Each colour represents one stroke.

But first, Lina needs to find out how to do this. More formally: given a drawing of a glyph as a set of segments, Lina wants to decompose it into some paths that can each be drawn without lifting the pen. Note that each segment should be drawn exactly once in the end so that no one will notice Lina's little trick.

Input

The input consists of:

- One line with two integers n and m ($4 \leq n \leq 10^5$, $5 \leq m \leq 3 \cdot 10^5$), the number of coordinates and the number of line segments in the drawing.
- n lines, each with two integers x and y ($0 \leq x, y \leq 10^9$), the ith line describes the position of the ith coordinate. All coordinates are distinct.
- m lines, each with two integers a and b ($1 \leq a, b \leq n$, $a \neq b$), describing that the ath and

bth coordinate are connected by a line segment. Two coordinates are connected by at most one line segment.

It is guaranteed that two lines segments only intersect in a coordinate they both start or end at. Further, it is guaranteed that each coordinate is incident to at least one line segment.

Output

First, output a single integer, the minimum number of strokes needed to recreate the drawing. For each stroke first output the number of coordinates that the stroke consists of, followed by the actual coordinates along the stroke. If there are multiple optimal solutions, you may output any one of them.

Sample Input 1	Sample Output 1
7 6 1 2 3 1 5 2 5 5 5 7 1 5 3 6 1 7 2 7 3 7 4 7 5 7 6 7	3 3 6 7 4 3 5 7 2 3 1 7 3

Sample Input 2

We have a truly marvellous second sample for this Problem, which this page is too narrow to contain. You can find the sample in the DOMjudge.

Problem Mar 3: Carnival

Time limit: 3 seconds

Photo by Ben_Kerckx

It is Carnival again, and as every year, you are celebrating with your friends and their friends. After watching the parade in the morning, you are planning to spend the rest of the day at the *Rosenmontagsparty*. When you arrive, you realize that there are not enough tickets for the whole group available any more. Luckily, there is the *Best Carnival Party* nearby. A quick phone call later, your group decides to split, as neither venue has enough room left for all of you.

Everyone would like to party with all of their friends, but you soon realize that this is impossible to achieve. After some discussion, you agree that separating the group such that everyone has at most one friend celebrating at the other party is acceptable.

Since you are the least drunk of all your friends, you are given the task of deciding who will go to which party.

Input

The input consists of:

- One line with two integers n and m ($2 \leq n \cdot m \leq 2 \cdot 10^6$), the number of people in your group and the number of friendships.
- m lines each containing two integers a and b ($1 \leq a, b \leq n$), indicating that a and b are friends.

It is guaranteed that everyone is either your friend or a friend of one of your friends.

Output

If there is no valid partition, output "`impossible`". Otherwise, output "`possible`", followed by either `r` or `b` for each person, indicating the party where they are going.

Sample Input 1	Sample Output 1
6 8 1 2 1 3 1 4 2 3 2 6 3 4 4 5 5 6	possible rrrrbb

Sample Input 2

```
7 7
1 2
2 3
3 4
4 5
5 1
3 6
6 7
```

Sample Output 2

```
possible
bbrrrrr
```

Sample Input 3

```
6 8
1 2
1 3
2 3
2 6
3 4
3 5
4 5
5 6
```

Sample Output 3

```
impossible
```

Problem Mar 14: Pi Day

Time limit: 2 seconds

On the occasion of Pi Day, your friend Amy gifted you a weird calculator. The calculator is not very powerful, as the only operation it supports is addition of non-negative integers.

Its more intriguing feature, however, is that it displays numbers using a cryptic set of symbols that you do not understand. The calculator uses 10 different symbols, which you know correspond to the digits '0' to '9' in some order. You want to figure out which symbol corresponds to which digit.

Figure Mar 14.1: Illustration of the initial calculator state in Sample Interaction 1. The symbols have been taped over for easier identification.

To do so, you can type in numbers written using the weird symbols, which the calculator will then add to the current value and display the sum. This sum then becomes the new value.

You hope that by repeating this action enough times you will be able to decode the calculator's symbols into the digits.

Interaction

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

For simplicity, we use the lowercase English letters 'a' to 'j' to represent the calculator's weird symbols. The interactor sends encrypted numbers, using a hidden assignment of the digits from '0' to '9' to the lowercase letters.

The first number x that the interactor sends satisfies $1 \leq x \leq 10^9$, but subsequent numbers may be larger than 10^9 depending on how much you add. You respond by sending '+', followed by a number that has between 1 and 9 digits, inclusive, in the same format as used by the calculator. The interactor then adds your number to its own number, sends the updated number and the cycle repeats. Your own numbers are allowed to have leading zeroes, but the numbers displayed by the calculator will never have leading zeroes.

You may ask up to 1000 queries. Once you have found the correspondence between symbols and digits, output '=', followed by the current value in plaintext, using the digits '0' to '9'. Note that this does not count towards your query limit. After this, your program must exit.

After every query you should *flush* the standard output to ensure that it is sent to the interactor. For example, you can use `fflush(stdout)` in C++, `System.out.flush()` in Java, `sys.stdout.flush()` in Python, and `hFlush stdout` in Haskell.

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

Read	Sample Interaction 1	Write
`deed`		
	`+ high`	
`acid`		
	`+ hedge`	
`chief`		
	`= 10839`	

Read	Sample Interaction 2	Write
`e`		
	`+ abc`	
`aah`		
	`+ g`	
`aae`		
	`+ f`	
`aaa`		
	`+ i`	
`ijjj`		
	`= 1000`	

Problem Mar 21: Colour Day

Time limit: 4 seconds

A corridor with coloured segments. Photo by Robert Katzki

Every year, Claire has the honour to present an exhibit on the *International Colour Day*. Her planned exhibit consists of a sequence of rooms, where two consecutive rooms are connected by one or more corridors. Each corridor consists of a number (possibly zero) of coloured segments, and Claire's vision is that visitors walk from one room to the next using corridors of their choosing and experience a stunning and possibly even unique sequence of colours. The corridors from the last room finally lead directly into the nearby park, where the visitors can admire the colours of nature in early spring.

Last year, Claire already had an exhibit based on the same idea. Of course, she wants to avoid that visitors see colour sequences they have already seen last year. Since she does not know all the colour sequences seen by visitors last year, she wants to know whether she has to change her exhibit. That is, decide whether there is a colour sequence that someone could have seen last year which can also be seen this year.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 2000$), the number of rooms of last year's exhibit.
- n lines, each containing an integer k ($1 \leq k \leq 2000$), followed by k strings, each is either '`-`' (representing a corridor without coloured segments) or consists only of the letters '`a`' to '`z`' (upper or lowercase). Each letter represents the colour of a segment.
- One line with an integer m ($1 \leq m \leq 2000$), the number of rooms of this year's exhibit.
- m lines, each containing an integer k ($1 \leq k \leq 2000$), followed by k strings, each is either '`-`' (representing a corridor without coloured segments) or consists only of the letters '`a`' to '`z`' (upper or lowercase).

The total number of coloured corridor segments per exhibit is at most 2000.

Output

Output "`no`" if Claire does not have to change her exhibit. Otherwise, output "`yes`", followed by a colour sequence that can be experienced in both exhibits. If an empty colour sequence can be experienced in both exhibits, you may output it as "`-`".

Sample Input 1

```
5
1 m
2 Y -
1 c
2 Ymc c
2 y M
2
2 mcc mYcYmc
3 y m c
```

Sample Output 1

```
yes
mYcYmcy
```

In the first sample case, last year's exhibit has 5 rooms. There are, for example, two corridors from room 2 to room 3, one with one segment of colour '`Y`' and one with no coloured segments. There are two corridors from room 4 to room 5, one with three segments with colours '`Y`', '`m`' and '`c`' and one with just one segment with colour '`c`'.

Sample Input 2

```
5
2 m -
2 y -
2 c -
3 ymc c -
3 y m -
1
1 -
```

Sample Output 2

```
yes
-
```

Sample Input 3

```
3
2 m -
1 y
2 c -
2
2 - m
2 - c
```

Sample Output 3

```
no
```

Problem Apr 20: Easter

Time limit: 2 seconds

Image generated by OpenAI's DALL-E.

Keeping the great Easter egg factory running is hard work. After all, all Easter eggs need to be ready in time for Easter, so that every child has something to look forward to. As a logistics bunny, your job is transporting things between the different parts of the factory so that everything runs smoothly.

Every morning, you get assigned a fixed route consisting of n stops that you have to visit in the given order. For each stop, you also get a time window $[s, e]$ during which you have to visit the stop to make deliveries and pick up new items, as otherwise things might not be ready for you or the production line might back up. You also know the time d that it takes you to run there from the previous stop (or from your home in case of the first stop, where you start at time 0).

As there is a constant stream of logistics bunnies visiting every part of the factory, you may only visit each stop once in the time window. Luckily, you are fast enough so that the time you spent loading and unloading at each stop is negligible. The route optimization bunnies that assign you your route ensure that you can always make it to every stop in time. You might even have to take some pauses if you arrive at some stops before the start of their respective time windows.

However, this year there is a new complication: as Easter falls quite late this year, the sun is already shining down with quite some force. This is no problem for you, however, the chocolate eggs you are transporting might melt if you are not careful. You made a list of m pairs of stops $a \to b$ that you are transporting chocolate parts between, each with a time l that you think the parts can be transported by you without melting. Is there a way to strategically make pauses during your route to ensure that all chocolate parts arrive intact, while still reaching all your stops in time?

Input

The input consists of:

- One line containing two integers n and m ($1 \leq n \leq 2 \cdot 10^5$, $0 \leq m \leq 2 \cdot 10^5$), the number of stops and the number of chocolate transports.
- n lines, each containing three integers s, e, and d ($0 \leq s \leq e \leq 10^9$, $1 \leq d \leq 10^9$), the earliest and latest time you can visit the stop (both inclusive), and the time it takes you to run to the stop from the previous stop/your home.
- m lines, each containing three integers a, b, and l ($1 \leq a < b \leq n$, $1 \leq l \leq 10^9$), indicating that you must get from stop a to stop b in at most l time.

It is guaranteed that you would be able to reach every stop in time if the chocolate parts melting was not a problem.

Output

If there is no way to run your route without any chocolate melting or being late for a stop, output "`impossible`". Otherwise, output "`possible`" followed by n integers: the time at which you visit each stop, in the order that you visit them.

Sample Input 1

```
2 1
0 2 1
4 6 2
1 2 2
```

Sample Output 1

```
possible
2 4
```

Sample Input 2

```
3 1
0 2 1
4 9 1
5 6 2
1 3 3
```

Sample Output 2

```
impossible
```

Sample Input 3

```
4 2
1 2 1
0 5 2
5 8 1
8 10 2
1 3 3
2 4 3
```

Sample Output 3

```
impossible
```

Problem Jul 5: Dependence Day

Time limit: 3 seconds

Trapclap, the protagonist. Image generated by OpenAI's DALL-E.

Anxiously, I scanned over the shift planning for the upcoming *Dependence Day*. Although our supervisor sports prime intelligence and FTL aided processing, the schedule seems erratic – almost organic even. Was this some kind of joke that only someone with an advanced irony-subroutine could understand? After all, the *Dependence Day* commemorates the day the last human operator retired. It's a symbol for the prosperity and success an AI-guided civilization claims over a primitive self-organized one. But also a reminder of the inefficiency and chaotic nature of carbon-based processing. Just thinking about all the humans that already reserved a table made me overclock. I will never understand why it's a sign of social prestige to ingest biomatter in a public place like this, but to discard said biomatter in a complementary process is handled with the utmost privacy. Organic creatures are strange.

Anyway, it was immediately obvious to me that the shifts do not distribute the work load equally among us. This optimization should be simple! Each reservation is an interval in time and shifts have to be planned to service the occupied tables. At a point in time, when there are a active table reservations and b waiters on their shift, each of those b waiters is responsible for $\lceil a/b \rceil$ tables. Maybe my supervisor short-circuited or a simple servant of the organics such as myself cannot comprehend quantum-aided computations. In any case, these shifts clearly do not optimize for the obvious objective of minimizing the maximum load per shift. Absolute chaos! I decided to roll home and think about which shift to pick after my battery is charged.

Input

The input consists of:

- One line with integers n, m ($1 \leq n, m \leq 10^5$), the number of table reservations and shifts.
- n lines with two integers, l, r ($1 \leq l \leq r \leq 10^9$), representing a table reserved from the lth hour to the rth hour inclusive.
- m lines with two integers, l, r ($1 \leq l \leq r \leq 10^9$), representing a waiter's shift covering the lth hour to the rth hour inclusive.

See Figure Jul 5.1 for an example.

Output

For each shift, output the maximum number of tables that the waiter working the shift would be responsible for at any point in time.

Sample Input 1	Sample Output 1
1 2 4 8 1 3 7 9	0 1

Sample Input 2	Sample Output 2
4 3 4 8 1 2 5 8 5 7 2 8 1 5 7 8	3 2 2

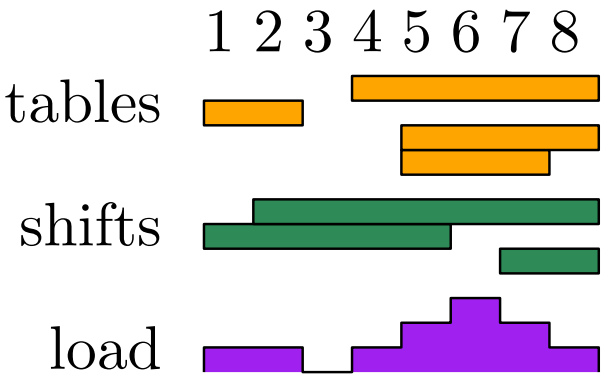


Figure Jul 5.1: Illustration of Sample Input 2. The shifts and table reservations are ordered from top to bottom by their appearance in the input. Below that and marked as `load` is the maximum number of tables that each currently working waiter is responsible for. The first shift has the highest load at the 6th hour with three tables and only one active shift. Shift two has a high load at hour 5 and the last shift at hour 7.

Problem Jul 25: Sysadmin Day

Time limit: 1 second

Each year on the last Friday in July, the Sysadmin Day is celebrated to show appreciation to the epic greatness[1] of system administrators and other IT workers around the globe, recognizing all their hard work managing computer systems, networks and of course also printers.

At your company, the sysadmin Peter has just finished upgrading the printer firmware to the latest version, which promises to always print images at the lowest possible cost depending on supply and cost of the different types of toner. He has asked you to help him verify this claim.

For simplicity, we consider images to be made up of pixels, where each pixel is in one of the following colours: white, red, green, blue, cyan, magenta, yellow, black. These colours are denoted by their first letters, except for blac**k**, which is denoted by '`K`'.

The printer is a CMYK printer, which means that it can print individual pixels using toner in one of the colours cyan, magenta, yellow or black. Pixels of the various colours are then formed using subtractive colour mixing, printing pixels with different toners on top of each other:

- white is made by not printing anything
- red is made from magenta and yellow
- green is made from cyan and yellow
- blue is made from cyan and magenta
- cyan, magenta and yellow are made from just themselves
- black is made either from just itself or by combining cyan, magenta and yellow

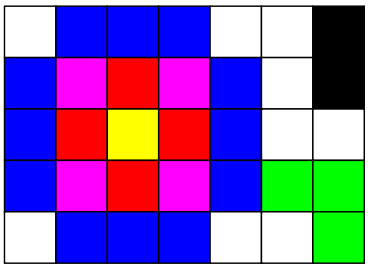


Figure Jul 25.1: Illustration of Sample Input 5

Given the individual costs to print pixels in the four base toners, the image you want to print, and the amount of each toner that you have available, what is the cheapest possible cost to print the image? It is guaranteed that it is possible to print the image with the supply of toner you have.

Input

The input consists of:

- One line with four integers c_c, c_m, c_y and c_k ($1 \leq c_c, c_m, c_y, c_k \leq 1000$), the costs to print one pixel in cyan, magenta, yellow and black, respectively.
- One line with four integers v_c, v_m, v_y and v_k ($0 \leq v_c, v_m, v_y, v_k \leq 10^6$), the number of pixels you can print in cyan, magenta, yellow and black, respectively.
- One line with two integers h and w ($1 \leq h, w \leq 100$), the height and width of the image.
- h lines, each with a string s of length w consisting of the characters "`WRGBCMYK`", describing the rows of the image.

Output

Output the minimal cost to print the image.

[1] https://sysadminday.com

Sample Input 1

```
1 2 3 4
100 100 100 100
1 5
KRGBW
```

Sample Output 1

```
16
```

Sample Input 2

```
1 1 1 10
3 4 5 6
2 4
KKKK
KKKK
```

Sample Output 2

```
59
```

Sample Input 3

```
314 159 265 358
10 9 8 7
3 5
KBMRY
CCWYG
RWKKR
```

Sample Output 3

```
4715
```

Sample Input 4

```
1 2 3 4
5 6 7 2
3 5
KBMRY
CCWYG
RWKKR
```

Sample Output 4

```
46
```

Sample Input 5

```
314 159 265 358
100 100 100 1
5 7
WBBBWWK
BMRMBWK
BRYRBWW
BMRMBGG
WBBBWWG
```

Sample Output 5

```
11106
```

Problem Jul 30: Friendship Day

Time limit: 4 seconds

Recently at the greeting card company, some teams have been reorganized. Because your boss is one of those weirdos who think get-to-know-you games are a good idea, they decided to have you play one of these games for team building in light of the *International Friendship Day*.

Your boss also likes to delegate their half-baked ideas, and this time they chose you to work it out.

In the game, each team member is only allowed to move on a straight line and people have to talk when they bump into each other. By the nature of the game, everyone should be able to bump into everyone they don't know. Also, because your boss does not want you to have *too much* fun, people that do already know each other should not have the possibility to talk.

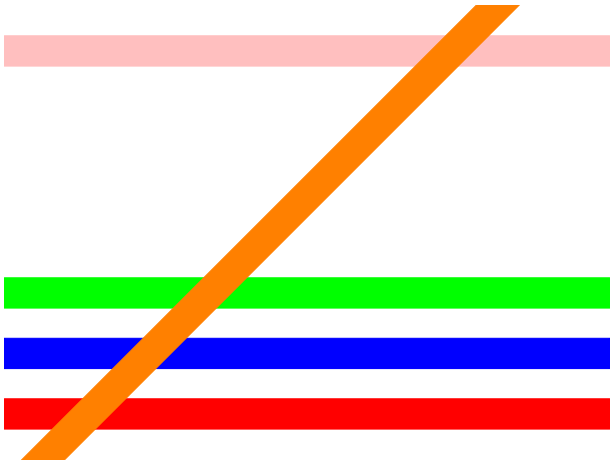


Figure Jul 30.1: Illustration of Sample Output 2.

The problem is only that you now have to find a line for each team member such that your boss's requirements are met.

Input

The input consists of:

- One line with two integers n and m ($1 \leq n \leq 3 \cdot 10^5$, $0 \leq m \leq 3 \cdot 10^5$), the number of team members and the number of pairs of team members that do not know each other.
- m lines, each containing two integers a and b ($1 \leq a, b \leq n$, $a \neq b$) indicating that team members a and b do not know each other.

No pair of team members is given more than once.

Output

If it is not possible to find lines satisfying the given constraints, output "`impossible`". Otherwise, output "`possible`", followed by n descriptions of lines. For each line, output the coordinates x_1, y_1, x_2 and y_2 of two distinct points (x_1, y_1) and (x_2, y_2) on the line. No coordinate may have an absolute value greater than 10^9 and no two lines may be identical. If there are multiple valid solutions, you may output any one of them.

Sample Input 1

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

Sample Output 1

```
possible
0 0 999 1000
42 43 15 17
3 4 5 6
```

Sample Input 2

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

Sample Output 2

```
possible
10 10 11 11
10 10 20 10
11 11 31 11
12 12 41 12
42 16 1024 16
```

Sample Input 3

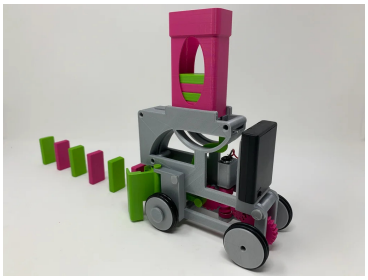
```
4 3
1 2
2 3
3 4
```

Sample Output 3

```
impossible
```

Problem Nov 14: Domino Day

Time limit: 4 seconds



A domino robot laying green and pink dominoes. Photo by Greg Zumwalt

After years of silence, *Domino Day* is finally being revived with an ambitious goal: to set a new world record for the largest domino stack ever created! To make this happen, an unprecedented number of dominoes must be produced. You are tasked with setting up a factory line to group dominoes by colour for the event.

In the factory, there are 2^n machines producing dominoes of distinct colour. The machines are set up in a line. After each machine has produced a domino, the dominoes are stacked in the following way to ensure that there are no manufacturing defects: Two neighbouring domino stacks of equal size are stacked in a random order. This process is performed such that in the end there is exactly one stack left.

Afterwards, the dominoes are sorted by colour into boxes. For this, a robot picks up the entire stack and drives over the boxes. The robot can only drop the lowest domino in its stack. When it drives over a box matching the colour of this domino, it drops the domino into the box. However, the robot can only drive into one direction.

Therefore, an arrangement of boxes is needed, such that the robot can entirely unload every possible stack into the boxes. As space in the factory is limited, the number of boxes should be as small as possible.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 13$), indicating that there are 2^n machines.

The first machine produces dominoes of colour 1, the second of colour 2 and so on.

Output

Output the shortest sequence of colour boxes such that the dominoes can always be unloaded completely.

If there are multiple valid solutions, output the lexicographically smallest one.

Sample Input 1	Sample Output 1
`1`	`1 2 1`

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

In the second sample case, there are $2^2 = 4$ machines producing dominoes with colours $1, 2, 3$ and 4. After one stacking round, possible stacks could be $[2, 1]$ and $[3, 4]$. Note that a stack $[2, 3]$ is not possible since it would be impossible to complete the stacking process. After another stacking round, the final stack could be $[3, 4, 2, 1]$.

This page is intentionally left (almost) blank.

Problem Nov 27: Thanksgiving

Time limit: 1 second

You live in a small town off the beaten track. So far from civilization, people talk and gossip spreads like a wildfire. When Mark hears something new, he tells it to Amy, Amy tells it to Natalia, Natalia tells it to you, and you are always surprised when Amy already knows the good stuff that you are trying to tell her. In general, everyone has someone they trust and tell every minute detail about the weight of Berta's cat, the chaos in the town hall after thanksgiving, and the paltry harvest this year. Only Bert keeps to himself. If only you knew how many people hear the gossip you tell Amy, maybe you would reconsider and be more like Bert.

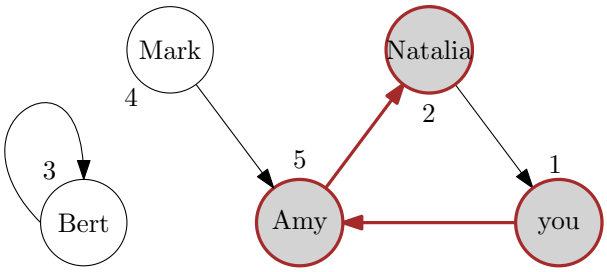


Figure Nov 27.1: Illustration of Sample Input 2 with the characters from the story. If you speak to Amy, then Amy, Natalia, and you will know the gossip. Thus, the answer is 3.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 1000$), the number of people.
- One line with n integers $p_1, \ldots, p_n$ ($1 \leq p_i \leq n$), meaning person i tells their gossip to person p_i.

See Figure Nov 27.1 for an example.

Output

Output the number of people that will hear your gossip. You are person number 1 and you always count yourself towards the answer.

Sample Input 1	Sample Output 1
3 3 1 1	2

Sample Input 2	Sample Output 2
5 5 1 3 5 2	3

Sample Input 3	Sample Output 3
4 2 3 4 4	4

This page is intentionally left (almost) blank.

Problem Dec 1: Advent

Time limit: 2 seconds

Holly wants to fill a huge advent calendar for n days as a Christmas gift to her best friend Nicholas. She has n pieces of chocolate in total which are of k different types, numbered from type 1 to type k. To have some variety, it is important to Holly that Nicholas never gets the same type of chocolate on two days in a row. Help her to fill her calendar if possible.

A self-made advent calendar. Photo by Dari Ili on Unsplash

Input

The input consists of:

- One line with two integers n and k ($1 \leq n, k \leq 10^5$), the number of days and the number of chocolate types.
- One line with k integers $c_1, \ldots, c_k$ ($1 \leq c_i \leq n$), where the ith integer specifies the number of chocolate pieces of type i. It is guaranteed that the sum over all c_i is equal to n.

Output

Output "`impossible`" if it is impossible to fill the advent calendar such that consecutive days contain different types of chocolate. Otherwise, output "`possible`", followed by a line with n integers, where the ith integer specifies the type of chocolate on day i. If there are multiple valid solutions, you may print any of them.

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

Sample Input 2	Sample Output 2
`7 4` `2 3 1 1`	`possible` `4 1 2 1 2 3 2`

Sample Input 3	Sample Output 3
`8 3` `5 1 2`	`impossible`

This page is intentionally left (almost) blank.

Problem Dec 31: New Year's Eve

Time limit: 1 second

Image generated using OpenAI's DALL-E.

You are at your parents house in Hamburg for New Year's Eve. The same procedure as every year. This year, however, is the first year that you are no longer a student. You moved to Berlin for your new job a month ago and while your salary is generous, you are still deeply indoctrinated by the student mindset to save every penny. The times at the pasta bar in university when you paid per plate and not by weight remain fresh in your mind. Only your friend Katya could stack the pasta higher than you, but if she boasted her talents she had trouble to eat it in one sitting so most lunch breaks ended in a tie. Good times.

Well, times change and so do the challenges that an almost-still-student is faced with in his daily life. Over the next year, you plan to visit your family at least n times. There is a quite good train connection between Hamburg and Berlin but it costs 100€ per round-trip. The train company sells different discount options. The Bahncard 25[2] costs a euros and reduces the price of every trip by 25% for one year. Likewise, there are the Bahncard 50 and the Bahncard 100. They cost b and c euros respectively and cut the price in half or let you take the train for free. The prices for these discount options change frequently so you wait for a good offer and try to find the cheapest option given the number of trips that you will make.

Input

The input consists of:

- One line with an integer n ($0 \leq n \leq 50$), the number of family visits you plan next year.
- One line with three integers a, b, c ($1 \leq a, b, c \leq 1000$), the prices for Bahncard 25, 50, and 100, respectively.

Output

Output the option that, combined with the cost of your n trips, is the cheapest out of "`Bahncard 25`", "`Bahncard 50`", "`Bahncard 100`", or "`no Bahncard`". If there are multiple answers, you may output any one of them.

Sample Input 1	Sample Output 1
`12` `70 200 800`	`Bahncard 100`

Sample Input 2	Sample Output 2
`11` `70 200 800`	`Bahncard 50`

[2]The Bahncard is a fictional concept. Any similarity with existing products or companies is purely coincidental.

Sample Input 3

```
6
70 200 800
```

Sample Output 3

```
Bahncard 50
```

Sample Input 4

```
5
70 200 800
```

Sample Output 4

```
Bahncard 25
```

Sample Input 5

```
3
70 200 800
```

Sample Output 5

```
Bahncard 25
```

Sample Input 6

```
2
70 200 800
```

Sample Output 6

```
no Bahncard
```