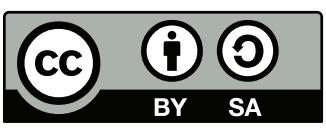
Freshmen Programming Contest 2026

Problems

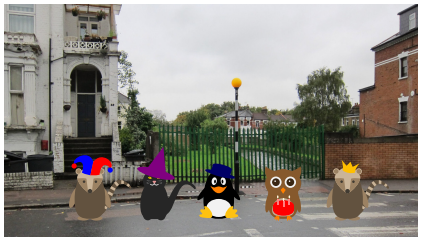
A	Animal Attire
B	Building Beaver
C	Coven Complications
D	Digit Display
E	Ernest's Endeavour
F	Ford's Funny Fields
G	Game Night Groups
H	Hasty Hiker
I	Intricate Idioms
J	Joker Juggling
K	Kracked Enkoder
L	Labyrinth Obstacle Layout



A Animal Attire

Time limit: 1s

The Coati Queen has decided to make the animal parade a recurring tradition, to be held n more times during her reign. The tradition will now require all participants to wear a different outfit for each occurrence of the parade. An outfit consists of exactly k types of clothing. For each type, there can be many different individual clothing items. Each clothing item is of exactly one type. Two outfits are different, if they have a different item in at least one type of clothing.



CC BY-SA 2.0 by Alan Stanton on Flickr

Adélie the Penguin has decided she will be smart about it, and buy all the items upfront to receive a good discount. What is the least total number of different clothing items she needs to own?

As an example, consider the second sample input. The outfits consist of 3 types of clothing: for example, a shirt, a pair of trousers, and a hat. To make the 8 different outfits that she requires, Adélie can make combinations of 2 differently patterned shirts, 2 differently coloured pairs of trousers, and 2 different hats, for a total of 6 clothing items.

Input

The input consists of:

- One line with two integers k and n ($1 \leq k \leq 100$, $1 \leq n \leq 10^{12}$), the number of different types of clothing and the number of upcoming parades.

Output

Output the least total number of different clothing items Adélie needs to own, in order to wear a different outfit for each of the upcoming parades.

Sample Input 1	Sample Output 1
2 4	4

Sample Input 2	Sample Output 2
3 8	6

Sample Input 3	Sample Output 3
4 30	10

This page is intentionally left blank.

B Building Beaver

Time limit: 3s

Beaver Bert is an expert in everything trees: aspen trees, willow trees, poplar trees, birch trees, and *binary search trees* (BSTs). Most of these trees are useful for building dams, but to keep track of the inventory of all felled trees for his dam, Bert uses a BST. A BST is a binary tree with a value in each node. These values satisfy the BST property: the value at each node is strictly larger than all values in its left subtree and strictly smaller than all values in its right subtree.

CC BY-SA 2.0 by Steve on Wikimedia Commons

A simple BST insertion, given a value x that does not yet appear in the tree, goes as follows:

- Start at the root.
- If the value of the current node is smaller than x, recurse to the right subtree. Otherwise, recurse to the left subtree.
- If this subtree does not exist, insert a new node at this position with value x.

The algorithm does not perform rotations to rebalance the tree.

As Beaver Bert already felled some trees, he wants to add them to his inventory. For simplicity, he gives a unique identifier between 1 and n to each of his n felled trees. Beaver Bert adds the felled tree identifiers to the BST one by one by using simple BST insertion by choosing an order $p_1, \ldots, p_n$ of the numbers 1 to n. Specifically, he makes p_1 the root, and then inserts $p_2, p_3, \ldots, p_n$ into the simple BST one by one.

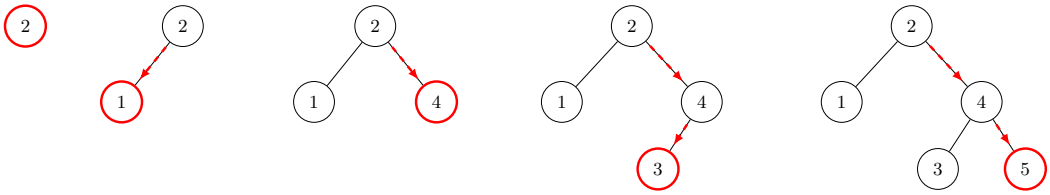


Figure B.1: Example for inserting the values in order $(2, 1, 4, 3, 5)$ in a BST, corresponding to the second sample output.

It would be really nice if, after this process, the BST satisfies the AVL balance property,[1] so Busy Beaver Bert has no problems with looking up specific felled trees in his inventory fast. Help Bert by finding the lexicographically smallest order[2] of the felled tree identifiers $1, \ldots, n$ that makes the BST satisfy the AVL balance property.

[1]The AVL balance property of a binary tree states that at each vertex, the difference between the two heights of its children's (possibly absent) subtrees is at most 1. Note that the height of an absent subtree is equal to 0.

[2]A list of numbers $p_1, p_2, \ldots, p_n$ is lexicographically smaller than $q_1, q_2, \ldots, q_n$, if on the first index f where p and q differ, $p_f < q_f$.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 2 \cdot 10^5$), the desired size of the BST.

Output

Output an ordering of the numbers 1 to n, $p_1, \ldots, p_n$, the lexicographically smallest order of the felled trees such that, when inserted into a simple BST in that order, this BST satisfies the AVL property.

It can be proven that some order exists such that the AVL property holds.

Sample Input 1	Sample Output 1
1	1

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

C Coven Complications

Time limit: 2s

After some unexpected circumstances, you have become the ruler of the most prestigious witch coven in the forest. This coven controls some of the witch communities in the forest – specifically, the ones with no ferrets. The communities are linked by portals: each portal links exactly two communities.

Do not get fooled by their cute appearance, the FPCs are here to drain power from your witches! Ferret: CC BY-SA 4.0 by Alfredo Gutiérrez on Wikimedia Commons Hat: CC BY-SA 3.0 by WarX on Wikimedia Commons

Unfortunately, all communities owned by rival witch covens are swarmed with Ferret Potion Creators (FPCs). Each community i controlled by a rival contains $f_i > 0$ ferrets, and each day, the ferrets of this community create potions that drain power from f_i witches of each linked community you own.

After years of studying spells, you have finally learned a spell that can permanently protect a community from all FPC attacks. However, the spell is exhausting and can only be cast once per day. Each day proceeds as follows:

1. You cast the sealing spell on one of your unsealed communities. Witches in this community are immediately protected and lose no power from this day onwards.
2. Each of your own remaining unsealed communities loses a number of witches equal to the total number of ferrets in their directly linked rival communities.

Calculate the minimum number of witches that will lose their power before all of your communities are sealed, so you know how many shamans to hire to heal them. You may assume that each of your communities has an unlimited supply of witches.

Input

The input consists of:

- One line with two integers n and m ($1 \leq n \leq 10^5$, $0 \leq m \leq 10^5$), the number of communities and the number of portals between communities.
- One line with n integers $f_1, \ldots, f_n$ ($0 \leq f_i \leq 10^9$ for each i), the number of ferrets in each community. Your witch coven owns a community if and only if $f_i = 0$.
- m lines, each with two integers u and v ($1 \leq u < v \leq n$), indicating that a portal links the two communities u and v. There is at most one portal between the same two communities. There are no portals from a community to itself.

Output

Output the minimum number of witches that lose their power before all of your communities are sealed.

Sample Input 1

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

Sample Output 1

```
10
```

Sample Input 2

```
7 9
20 15 0 2 0 10 0
1 2
1 4
1 5
2 7
3 4
3 5
3 6
5 6
6 7
```

Sample Output 2

```
49
```

D Digit Display

Time limit: 1s

After Olimar's departure from PNF-404, the Pikmin became bored and lonely. Wanting to impress Olimar on his potential return, and knowing how mathematically inclined Olimar is, they find a 7-segment display. Upon learning what its symbols represent, they think the best way to impress Olimar would be to represent the biggest number possible with their bodies on the ground, in hopes that Olimar will see it from his spaceship and return.

Captain Olimar. © Nintendo via the Pikmin Wiki, fair use

Each Pikmin can form exactly one segment by placing their body on the ground. The space on the ground is unbounded. Given the number of Pikmin available to help, what is the largest possible number that their bodies can form?

Number of segments:	6	2	5	5	4	5	6	3	7	6

Figure D.1: A visualization of how each of the digits 0–9 are displayed on a 7-segment display, with the number of segments required.

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 1000$), the number of available Pikmin.

Output

If it is possible for the Pikmin to form a number with their bodies on the ground using exactly n segments, output the largest possible number. Otherwise, output "`impossible`".

Sample Input 1	Sample Output 1
`1`	`impossible`

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

This page is intentionally left blank.

E Ernest's Endeavour

Time limit: 3s

Jack, Being Earnest. CC BY-SA 2.0 by Otterbein University Theatre & Dance on Flickr

Jack Worthing lives a dangerous double life. In the countryside, he is known as Jack. Meanwhile, in London, he is known as Ernest. He is terrified that his two identities will be discovered. To analyze this, he maps the residents of London and its surroundings as a graph of n people.

Each person in the network either knows him as "Jack", knows him as "Ernest", or does not know him. Additionally, the residents of London are known to be quite social: we know there are a total of m friendships between them.

Because rumours travel fast in 19th-century England, Jack will be exposed if a person who knows him as "Jack" can communicate with a person who knows him as "Ernest". Two people can communicate if they know each other directly, or if they are connected by a sequence of mutual friendships (that is, there is a path between them in the graph).

Help Jack figure out if he is safe, or if he needs to flee to Paris immediately.

Input

The input consists of:

- One line with two integers n and m ($1 \leq n \leq 3 \cdot 10^5$, $0 \leq m \leq 6 \cdot 10^5$), the number of people and the number of friendships.
- One line with n integers $t_1, \ldots, t_n$ ($t_i \in \{0, 1, 2\}$ for each i).
  - If $t_i = 0$, person i does not know him.
  - If $t_i = 1$, person i knows him as "Jack".
  - If $t_i = 2$, person i knows him as "Ernest".
- m lines, each with two integers u and v ($1 \leq u < v \leq n$), indicating that person u and person v know each other directly. Each friendship is given at most once.

Output

If Jack is safe, output "`safe`". If Jack is in danger, output the indices of two people that know Jack by different names and can communicate.

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

Sample Input 1

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

Sample Output 1

```
safe
```

Sample Input 2

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

Sample Output 2

```
3 2
```

Sample Input 3

```
5 4
1 1 0 2 2
1 2
2 3
3 4
4 5
```

Sample Output 3

```
2 4
```

F Ford's Funny Fields

Time limit: 3s

After learning about SEP ("Somebody Else's Problem") Fields, Arthur Dent wants to use them to hide his spaceship.

Inspired by Slartibartfast, his ship is similar to a Subway restaurant – that is, long and narrow. More precisely, the ship has width 1 and length s.

To hide his ship, Arthur borrows SEP Fields of different costs and sizes from Ford. Of each type, Ford owns x SEP Fields of width 1 and length ℓ that cost c Altarian Dollars. As Ford's disposition is hardly predictable, Arthur wants to minimize the cost of the SEP Fields he needs to ask Ford to borrow.

You are tasked with helping Arthur compute the minimum cost (in Altarian Dollars) in order to hide his entire ship. The total length of the borrowed SEP Fields is allowed to be larger than the length of the ship.

Surely, nobody will notice this spaceship hidden by SEP Fields. CC BY-SA 2.5 by Nathan Beach on Wikimedia Commons

Input

The input consists of:

- One line with two integers n and s ($1 \leq n \leq 100$, $1 \leq s \leq 2000$), the number of different field types Ford has, and the length of Arthur's ship.
- n lines, each with three integers x, ℓ, and c ($1 \leq x \leq 20$, $1 \leq \ell \leq 100$, $1 \leq c \leq 10^9$), representing a type of SEP Field: x is the number of such fields Ford has, ℓ is the length of such a field, and c is the cost of such a field.

Output

If it is possible to hide the ship, output the minimum total cost to do so. Otherwise, output "`impossible`".

Sample Input 1	Sample Output 1

`2 10`
`3 4 3`
`2 3 1`
	`5`

Sample Input 2	Sample Output 2
`1 10` `3 3 10`	`impossible`

Sample Input 3	**Sample Output 3**
2 10 2 6 5 1 12 8	8

G Game Night Groups

Time limit: 1s

You are hosting a game night with many friends, where each table has a group playing Fables & Perilous Caverns (FPC). FPC is a game that can be played with at least x and at most y players. To make the game night start smoothly as soon as all your friends have arrived, you want to know beforehand whether they can be split into groups such that each group can play FPC.

As an example, consider the second sample input, where groups should have between 5 and 7 players. If you would have 18 friends, you could split them into three groups, as visualized in Figure G.1. It can be shown that no valid arrangement exists for 9 people. With strictly more than 9 people, it is always possible to host the game night where everybody can play FPC.

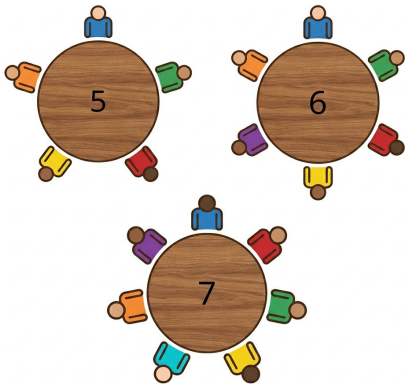


Figure G.1: A possible arrangement for 18 people when $x = 5$ and $y = 7$. Image generated using Gemini.

You want to make sure that you invite enough people so that nobody is left out. What is the largest number of friends that can *not* be split into groups with the required sizes to play FPC?

Input

The input consists of:

- One line with two integers x and y ($2 \leq x < y \leq 10^6$), indicating that the groups must have at least x and at most y people.

Output

Output the largest number of people that can *not* be split into groups with these size constraints.

It is guaranteed that under the input constraints such a number always exists.

Sample Input 1	Sample Output 1
4 6	7

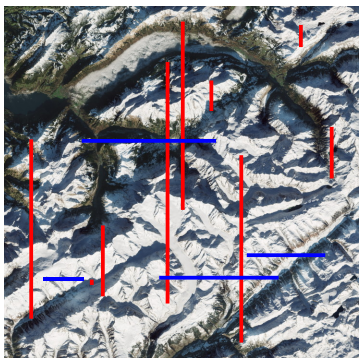
Sample Input 2	**Sample Output 2**
5 7	9

Sample Input 3	**Sample Output 3**
67 69	2210

H Hasty Hiker

Time limit: 3s

You are embarking on a hiking journey through the famous Dutch mountainside. When viewed from above, the mountainside can be seen as a two-dimensional plane. On this plane, there are n horizontal or vertical roads. That is, the two endpoints of each road have either the same x-coordinate, or the same y-coordinate. Furthermore, it is guaranteed that there are no roads of length 0. Lastly, it is guaranteed that two horizontal roads never intersect, and likewise, two vertical roads never intersect.

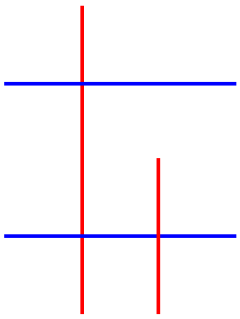


A mountain range with horizontal and vertical roads. CC BY-SA 3.0 IGO by ESA on www.esa.int, modified

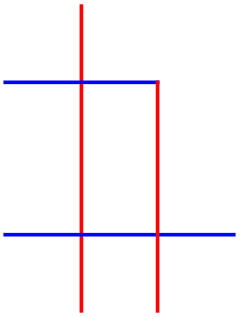
We call an ordered quadruple of roads (r_1, r_2, r_3, r_4) a *beautiful trail*, if the following conditions hold:

- all four roads r_1, r_2, r_3 and r_4 are *pairwise distinct*.
- r_1 and r_2 intersect.
- r_2 and r_3 intersect.
- r_3 and r_4 intersect.

Calculate the number of *beautiful trails* you can form with the roads given in the input.



The answer for the first sample input is 2, since there are two ways in which the roads can be ordered to form a *beautiful trail*.



The answer for the second sample input is 8, since there are 4 roads you can start on, and you can hike either clockwise or counterclockwise.

Input

The input consists of:

- One line with an integer n ($4 \leq n \leq 5000$), the number of roads.
- n lines, each with four integers x_1, y_1, x_2, and y_2 ($|x_1|, |y_1|, |x_2|, |y_2| \leq 10^9$), indicating there is a road between (x_1, y_1) and (x_2, y_2).
  It is guaranteed that either $x_1 = x_2 \wedge y_1 < y_2$ or $x_1 < x_2 \wedge y_1 = y_2$.

It is guaranteed that two horizontal roads never intersect, and likewise, two vertical roads never intersect.

Output

Output the number of *beautiful trails*.

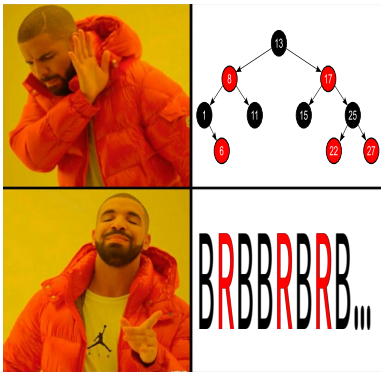
Sample Input 1	Sample Output 1
4 0 -3 0 1 -1 0 2 0 -1 -2 2 -2 1 -3 1 -1	2

Sample Input 2	Sample Output 2
4 0 -3 0 1 -1 0 1 0 -1 -2 2 -2 1 -3 1 0	8

I Intricate Idioms

Time limit: 2s

Ilya is a first-year Computer Science student, but more importantly, he is a language enthusiast. He has a very large collection of idioms that he spent years procuring. However, this collection has started to get out of hand, so Ilya is now looking to organize it. Recently, during the Algorithms & Data Structures lecture, he heard about some binary trees which are coloured in two colours, red and black. He left the lecture quite confused, but very interested about the topic, looking to find some potential uses for them while organizing his collection. Later, while trying to understand the red-black property, he converted all the trees to strings. However, something went wrong with his translation and he is now looking for your help to tell him how many steps he needs to convert them correctly.



Ilya after converting the trees to strings.
Internet meme: "Drakeposting", fair use
Tree: CC BY-SA 4.0 by Nomen4Omen on Wikimedia Commons

You are given n strings consisting of the characters '`R`' (Red) and '`B`' (Black). You need to make sure that the red-black property is satisfied for all strings, in as few moves as possible. In one move, you can replace one character from '`R`' to a '`B`' (but not vice versa). Ultimately, the following properties need to be satisfied:

- The leftmost character of each string is a '`B`'.
- In each string, there can be no two '`R`' characters adjacent to one another.
- The number of '`B`' characters in each string needs to be the same as the number of '`B`' characters in every other string.

What is the minimum number of moves to make the strings satisfy all properties?

Input

The input consists of:

- One line with an integer n ($1 \leq n \leq 2 \cdot 10^5$), the number of strings.
- n lines, each with a string s ($1 \leq |s| \leq 2 \cdot 10^5$), each character being either '`B`' or '`R`'.

The sum of lengths over all strings is at most $2 \cdot 10^5$.

Output

If it is possible to make the strings satisfy all properties, output the minimum number of moves needed to do so. Otherwise, output "`impossible`".

Sample Input 1

```
1
RBBRRBBBR
```

Sample Output 1

```
2
```

Sample Input 2

```
3
BRBRR
BRB
RBB
```

Sample Output 2

```
3
```

Sample Input 3

```
2
BBB
RR
```

Sample Output 3

```
impossible
```

J Joker Juggling

Time limit: 1s

Jamie the Joker has two full-time jobs: they juggle balls at the circus by day and juggle *match patterns* by night. Obviously, this is not a very healthy life style. After a long day of juggling (both kinds), Jamie is exhausted, but has only one task left to do: determine whether a pattern matches a word.

The word and the pattern consist of English lowercase letters, and the pattern may also contain *jokers* (represented by an asterisk: '`*`'). The pattern is said to match the word if it is possible to replace each joker by *the same* string (possibly empty), such that the pattern becomes equal to the word.

A joker, juggling jokers. CC0 (public domain) on publicdomainvectors.org, modified

Input

The input consists of:

- One line with a string p ($1 \leq |p| \leq 3 \cdot 10^5$) only consisting of letters and jokers ('`*`'), the pattern.
- One line with a string s ($1 \leq |s| \leq 3 \cdot 10^5$) only consisting of letters, the word.

All letters in the pattern and the word are English lowercase letters (`a`–`z`).

Output

Output "`yes`" if the pattern matches the word, or "`no`" if it does not.

Sample Input 1	Sample Output 1
`ba**` `banana`	`yes`

Sample Input 2	Sample Output 2
`apple` `banana`	`no`

Sample Input 3	Sample Output 3
`cherry*` `cherry`	`yes`

Sample Input 4	Sample Output 4
`**` `ttt`	`no`

This page is intentionally left blank.

K Kracked Enkoder

Time limit: 1s

Encoding and decoding play important roles in many computer systems. Especially for Karl, whose computer can only store at most 29 bits! Not knowing how to deal with this, Karl has turned to you to find a way to encode a number so that it can be stored on his computer, and to decode the number as well.

Karl's Potato Computer.
Internet meme at imgflip, fair use

This is a *multi-pass* problem: your program will be invoked two times: one run for encoding, and one run for decoding.

In the first run, your program receives the string "`encode`", together with a positive integer $1 \leq x \leq 2^{30} - 2$. This is the number that Karl wants you to encode. Your program must then output a non-empty string, consisting of at most 29 zeros and ones.

In the second run, your program receives the string "`decode`", together with the output of the first run of your program. Your program must then output Karl's number x.

Input

The input consists of:

- One line with the action that your program needs to perform: either the string "`encode`" or "`decode`".
- If the action is "`encode`":
  - One line with a string number x ($1 \leq x \leq 2^{30} - 2$), the number Karl wants you to encode.
- If the action is "`decode`":
  - One line with a binary string b ($1 \leq |b| \leq 29$), only consisting of digits 0 and 1. This is the binary string you used to encode Karl's number x.

This is a multi-pass problem. For each test case, your program will be invoked two times. It is guaranteed that the first pass is a "`encode`" action, and that the second pass is a "`decode`" action.

This problem does *not* provide a testing tool.

Output

If the action is "`encode`", output a string b of length $1 \leq |b| \leq 29$, only consisting of digits 0 and 1.

If the action is "`decode`", output the original number x.

Sample Case 1

Input	**Pass 1**	**Output**
`encode` `123456`	`100110011001`	

Input	**Pass 2**	**Output**
`decode` `100110011001`	`123456`	

L Labyrinth Obstacle Layout

Time limit: 1s

You need to construct a Labyrinth Obstacle Layout (LOL), for the famous game Walk Guys. The LOL will be the centre of the brand new game mode Square-A-Gone, so it has to obey some strict properties to make the game fair and fun.

The LOL is a square grid of $n \times n$ cells with obstacles (denoted with '`#`') and walkable tiles (denoted with '`.`').

A player starts in the top-left cell, and the finish is located on the bottom right cell. Both of these cells must be walkable tiles. In Square-A-Gone, whenever they step on a walkable tile, this tile falls down and disappears. So a player trying to reach the exit can do the following: they start on the top-left tile, and repeatedly walk to a tile which shares an edge with the current tile vertically or horizontally. As soon as the player leaves a walkable tile, it will fall down and the player cannot use it anymore. As this is Walk Guys, jumping is not allowed, only walking. When the player reaches the finish, the game is over.

Output a LOL with top left cell and bottom right cells being walkable, such that there are exactly 2 ways for a player to reach the finish tile. Two ways are different if a player at some point in time walks in a different direction.

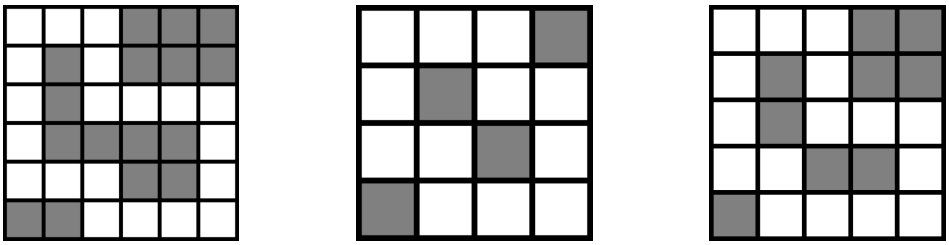


Figure L.1: Visualizations of the three sample outputs.

Input

The input consists of:

- One line with an integer n ($3 \leq n \leq 100$), the required length of a side of the square grid that makes up the Labyrinth Obstacle Layout.

Output

Output an $n \times n$ grid with characters '`#`' and '`.`' that is a valid Labyrinth Obstacle Layout.

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

Sample Input 1	**Sample Output 1**
6	...### .#.### .#.... .####. ...##. ##....

Sample Input 2	**Sample Output 2**
4	...# .#.. ..#. #...

Sample Input 3	**Sample Output 3**
5	...## .#.## .#... ..##. #....