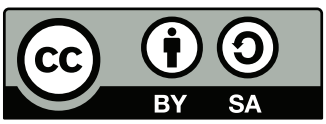
Udem Contest

Problems

A1	Associativity is optional (Hard)
A2	Associativity is optional (Easy)
B1	Boat Spawn (Hard)
B2	Boat Spawn (Easy)
C1	Choosing a pivot (Hard)
C2	Choosing a pivot (Easy)
D1	Dethroning the master (Hard)
D2	Dethroning the master (Easy)
E1	Euclid Worst Case (Hard)
E2	Euclid Worst Case (Easy)
F1	Fractions playing games (Hard)
F2	Fractions playing games (Easy)
G1	Giraffe excursion (Hard)
G2	giraffeexcursion easy



A1 Associativity is optional (Hard)

Time limit: 1s

Alice just learned of the quaternion in her group theory class. She is a bit annoyed that the group isn't abelian. To fix this, she invents a new operation on the quaternion group, $*$.
The 8 elements of the quaternion, $Q_8 := \{1, -1, i, -i, j, -j, k, -k\}$ multiply in the following way.
$\forall x \in Q_8, \forall y \in \{i, j, k\}$

1. $1 * x = x$
2. $-1 * x = x * -1 = -x$
3. $y * y = -1$
4. $i * j = j * i = k$
5. $i * k = k * i = j$
6. $j * k = k * j = i$

This new operation makes the elements of the quaternion commute with each other. However, the operation isn't associative (an operation is associative if the order of each operation doesn't matter). For example,
$i * (i * j) = i * k = j$
$(i * i) * j = -1 * j = -j$
Alice starts with a first expression, moves the bracket around, and obtains a second expression. She needs your help to figure out by what the expression has changed.

Input

You are given two strings on two lines representing the two expressions. Both expressions are made of parentheses and elements of Q_8. Both expressions are identical up to the parentheses. Both expressions are mathematically correct. Both strings have their lengths bounded by $4 \cdot 10^6$.

Output

Output $x \in Q_8$ such that
first expression $= x \cdot$ (second expression)

Sample Input 1	Sample Output 1
`i(ij)` `(ii)j`	`-1`

This page is intentionally left blank.

A2 Associativity is optional (Easy)

Time limit: 1s

Alice just learned of the quaternion in her group theory class. She is a bit annoyed that the group isn't abelian. To fix this, she invents a new operation on the quaternion group, $*$.
The 8 elements of the quaternion, $Q_8 := \{1, -1, i, -i, j, -j, k, -k\}$ multiply in the following way.
$\forall x \in Q_8, \forall y \in \{i, j, k\}$

1. $1 * x = x$
2. $-1 * x = x * -1 = -x$
3. $y * y = -1$
4. $i * j = j * i = k$
5. $i * k = k * i = j$
6. $j * k = k * j = i$

This new operation makes the elements of the quaternion commute with each other. However, the operation isn't associative (an operation is associative if the order of each operation doesn't matter). For example,
$i * (i * j) = i * k = j$
$(i * i) * j = -1 * j = -j$
Alice starts with a first expression, moves the bracket around, and obtains a second expression. She needs your help to figure out by what the expression has changed.

Input

You are given two strings on two lines representing the two expressions. Both expressions are made of parentheses and elements of Q_8. Both expressions are identical up to the parentheses. Both expressions are mathematically correct. Both strings have their lengths bounded by $4 \cdot 10^2$.

Output

Output $x \in Q_8$ such that
first expression $= x \cdot$ (second expression)

Sample Input 1	Sample Output 1
`i(ij)` `(ii)j`	`-1`

This page is intentionally left blank.

B1 Boat Spawn (Hard)

Time limit: 1s

Bob is coding a game. It has pirates, giraffes, and loads of fun.
The game is a $m \times n$ lattice with a procedurally generated island on it. The island is connected; that is, for every pair of points, there exists a sequence of inputs (up, down, left, right) to get from the first point to the second point without touching the water. Another quirk of his generation: every pair of points in the same row will be reachable by a path that only uses the input left and right.
Bob wants to spawn pirate ships at the furthest point (using the Manhattan distance) from any point on the island (to give enough time to the giraffes to defend themselves). Since the island is different on every playthrough and there might be more than one point that satisfies this condition, he is having trouble locating these points.

Input

On the first line, you are given two integers, m (length) and n (width), the dimensions of the lattice. $5 \leq$ m, n $\leq 10^5$ and m $\cdot$ n $\leq 10^8$.
The next m lines contain n 0's or 1's. The 0's represent a water tile where the pirate boats can spawn. The 1's represent an island tile. Every row contains at least one island tile.

Output

Output a pair of integers $(a, b) \in [0, n-1] \times [0, m-1]$, the coordinate of where to spawn the pirate boat. The coordinate $(0, 0)$ corresponds to the upper left corner. If there are multiple valid solutions, you may output any one of them.

Sample Input 1	Sample Output 1
5 5 1 1 1 0 0 0 1 1 1 0 0 0 0 1 1 0 0 0 0 1 0 0 1 1 1	0 3

This page is intentionally left blank.

B2 Boat Spawn (Easy)

Time limit: 1s

Bob is coding a game. It has pirates, giraffes, and loads of fun.
The game is a $m \times n$ lattice with a procedurally generated island on it. The island is connected; that is, for every pair of points, there exists a sequence of inputs (up, down, left, right) to get from the first point to the second point without touching the water. Another quirk of his generation: every pair of points in the same row will be reachable by a path that only uses the input left and right.
Bob wants to spawn pirate ships at the furthest point (using the Manhattan distance) from any point on the island (to give enough time to the giraffes to defend themselves). Since the island is different on every playthrough and there might be more than one point that satisfies this condition, he is having trouble locating these points.

Input

On the first line, you are given two integers, m (length) and n (width), the dimensions of the lattice. $5 \leq$ m, n $\leq 10^2$.
The next m lines contain n 0's or 1's. The 0's represent a water tile where the pirate boats can spawn. The 1's represent an island tile. Every row contains at least one island tile.

Output

Output a pair of integers $(a, b) \in [0, n-1] \times [0, m-1]$, the coordinate of where to spawn the pirate boat. The coordinate $(0, 0)$ corresponds to the upper left corner. If there are multiple valid solutions, you may output any one of them.

Sample Input 1	Sample Output 1
5 5 1 1 1 0 0 0 1 1 1 0 0 0 0 1 1 0 0 0 0 1 0 0 1 1 1	0 3

This page is intentionally left blank.

C1 Choosing a pivot (Hard)

Time limit: 1s

Pivot!

(Ross — Friends, season 5, episode 16)

Such a motivating and inspiring quote. So much so that Charlie decided to pivot his life in order. He wonders, how many pivots does he need to do? How big are the pivots? Could some pivots be better than others? None of his questions can be deduced from the quote.
He decides to learn more about another structure that uses pivots to get itself in order, an array of numbers.

```
def pivotSort(arr: [int]):
    if len(arr)<=1:
        return arr
    pivot=choosingPivot(arr)
    smaller, equal, larger = [], [], []
    for elem in arr:
        if elem<pivot:
            smaller.append(elem)
        elif elem>pivot:
            larger.append(elem)
        else:
            equal.append(elem)
    return pivotSort(smaller)+equal+pivotSort(larger)
```

The choice of the pivot is a complicated process. Charlie needs your help to pick the smallest pivot and still get the list in order. In other words, minimize $\sum_{p\in \mathrm{P}} p$, where P is a set of pivots that does order the array using the algorithm above.

Input

On the first line, you are given a positive integer $1 \leq n \leq 10^6$.
On the second line are n positive integers. The sum of every integer is bounded by $2 \cdot 10^9$.

Output

Let P be a set of pivots used to sort an array using the algorithm above. Find P such that $\sum_{p\in \mathrm{P}} p$ is as small has possible and output the value of this sum.

Sample Input 1	Sample Output 1
3 4 1 2	2

Sample Input 2	**Sample Output 2**
4 8 1 4 2	5

C2 Choosing a pivot (Easy)

Time limit: 1s

Pivot!

(Ross — Friends, season 5, episode 16)

Such a motivating and inspiring quote. So much so that Charlie decided to pivot his life in order. He wonders, how many pivots does he need to do? How big are the pivots? Could some pivots be better than others? None of his questions can be deduced from the quote.
He decides to learn more about another structure that uses pivots to get itself in order, an array of numbers.

```
def pivotSort(arr: [int]):
    if len(arr)<=1:
        return arr
    pivot=choosingPivot(arr)
    smaller, equal, larger = [], [], []
    for elem in arr:
        if elem<pivot:
            smaller.append(elem)
        elif elem>pivot:
            larger.append(elem)
        else:
            equal.append(elem)
    return pivotSort(smaller)+equal+pivotSort(larger)
```

The choice of the pivot is a complicated process. Charlie needs your help to pick the smallest pivot and still get the list in order. In other words, minimize $\sum_{p\in \mathrm{P}} p$, where P is a set of pivots that does order the array using the algorithm above.

Input

On the first line, you are given a positive integer $1 \leq n \leq 10^3$.
On the second line are n positive integers. The sum of every integer is bounded by $2 \cdot 10^9$.

Output

Let P be a set of pivots used to sort an array using the algorithm above. Find P such that $\sum_{p\in \mathrm{P}} p$ is as small has possible and output the value of this sum.

Sample Input 1	Sample Output 1
3 4 1 2	2

Sample Input 2	**Sample Output 2**
4 8 1 4 2	5

D1 Dethroning the master (Hard)

Time limit: 1s

You have been playing this game for years but still haven't won yet. Instead of changing your strategy of trusting the magic 8 ball, you have decided to write a computer program that plays for you.

The game consists of two integers, m and n. On your turn, you remove a strictly positive multiple of the smaller integer from the bigger integer. After which, it is your opponent's turn to do the same. The winner is the player that gets one of the integers to zero.

For example, say the two integers are 4 and 10.

You pick the smallest, in this case 4, and your possible moves are to remove $4 \cdot 1$ or $4 \cdot 2$ from 10. $4 \cdot 0$ is not strictly positive, and $4 \cdot 3 = 12$ is bigger than 10.

Say you choose to remove $4 \cdot 1$. The game state is now 4 6.

It is now your opponent's turn; his only move is to remove $4 \cdot 1$ from 6. The game state is now 4 2.

It is now your turn; you remove $2 \cdot 2$ from 4 and win the game.

Another important aspect of a great player is that they recognize when they are in a lost position. You would like your computer program to resign when it is impossible for you to win if both players play perfectly.

Interactivity

This is an interactive problem. The problem starts with two integers $1 \leq a, b \leq 10^{18}$. If you can't win with perfect play from the opponent, then output -1.

If you can always win with perfect play, then output a winning move. A move is the number of factors of the smallest number removed from the bigger number. For example, if you want to remove $1 \cdot 4$ from 6 then output 1.

After your move, another integer will be output. If the integer is -1, the opponent has resigned. Otherwise, the integer represents the move of your opponent.

Keep playing until the opponent resigns. When you win the game, the opponent will output -1.

This page is intentionally left blank.

D2 Dethroning the master (Easy)

Time limit: 1s

You have been playing this game for years but still haven't won yet. Instead of changing your strategy of trusting the magic 8 ball, you have decided to write a computer program that plays for you.

The game consists of two integers, m and n. On your turn, you remove a strictly positive multiple of the smaller integer from the bigger integer. After which, it is your opponent's turn to do the same. The winner is the player that gets one of the integers to zero.

For example, say the two integers are 4 and 10.

You pick the smallest, in this case 4, and your possible moves are to remove $4 \cdot 1$ or $4 \cdot 2$ from 10. $4 \cdot 0$ is not strictly positive, and $4 \cdot 3 = 12$ is bigger than 10.

Say you choose to remove $4 \cdot 1$. The game state is now 4 6.

It is now your opponent's turn; his only move is to remove $4 \cdot 1$ from 6. The game state is now 4 2.

It is now your turn; you remove $2 \cdot 2$ from 4 and win the game.

Another important aspect of a great player is that they recognize when they are in a lost position. You would like your computer program to resign when it is impossible for you to win if both players play perfectly.

Interactivity

This is an interactive problem. The problem starts with two integers $1 \leq a, b \leq 10^9$. If you can't win with perfect play from the opponent, then output -1.

If you can always win with perfect play, then output a winning move. A move is the number of factors of the smallest number removed from the bigger number. For example, if you want to remove $1 \cdot 4$ from 6 then output 1.

After your move, another integer will be output. If the integer is -1, the opponent has resigned. Otherwise, the integer represents the move of your opponent.

Keep playing until the opponent resigns. When you win the game, the opponent will output -1.

This page is intentionally left blank.

E1 Euclid Worst Case (Hard)

Time limit: 1s

The Euclidean algorithm is the following algorithm:

```
def euclid(a : int, b : int):
    n=0
    if a<b:
        a,b=b,a
    while b>0:
        n+=1
        r=a%b
        a=b
        b=r
    return b
```

The variable n counts the number of iterations of the Euclidean algorithm.

Input

The first and only line of the input contains a non-negative integer $1 \leq n \leq 10^6$.

Output

Find the smallest positive integer a such that there exists an integer b such that the number of iterations of the Euclidean algorithm with input a and b is n.
Since a will get quite big, output a mod $10^9 + 7$.

Sample Input 1	**Sample Output 1**
1	1

Sample Input 2	**Sample Output 2**
2	2

This page is intentionally left blank.

E2 Euclid Worst Case (Easy)

Time limit: 8s

The Euclidean algorithm is the following algorithm:

```
def euclid(a : int, b : int):
    n=0
    if a<b:
        a,b=b,a
    while b>0:
        n+=1
        r=a%b
        a=b
        b=r
    return b
```

The variable n counts the number of iterations of the Euclidean algorithm.

Input

The first and only line of the input contains a non-negative integer $1 \leq n \leq 16$.

Output

Find the smallest positive integer a such that there exists an integer b such that the number of iterations of the Euclidean algorithm with input a and b is n.
Since a will get quite big, output a mod $10^9 + 7$.

Sample Input 1	Sample Output 1
1	1

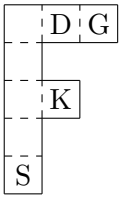
Sample Input 2	Sample Output 2
2	2

This page is intentionally left blank.

F1 Fractions playing games (Hard)

Time limit: 1s

Bob is coding a puzzle game. Bob is scared of making his game too hard, since the creator knows the solution to every puzzle. He has asked you to playtest one of the puzzles.



S is the starting location of the player. G is the goal the player needs to reach. However, the goal is locked behind a door D, for which the player will need to find a key K.
Being a programmer, you decide that the most efficient and straightforward way to solve this problem is to simulate random strings of inputs and check if the puzzle is solved.
Every input of the controller can be mapped to a digit in the following way.

- The ↑ input corresponds to 1 or 6. Move the player up 1 square. If the input makes the player cross a solid line, ignore the input.
- The → input corresponds to 2 or 7. Move the player right 1 square. If the input makes the player cross a solid line or a locked door, ignore the input.
- The ↓ input corresponds to 3 or 8. Move the player down 1 square. If the input makes the player cross a solid line, ignore the input.
- The ← input corresponds to 4 or 9. Move the player left 1 square. If the input makes the player cross a solid line, ignore the input.
- The pause/unpause input corresponds to 5 or 0. Switch the state of the game from unpause to pause or pause to unpause. Every movement input has no effect while the game is in the pause state.

All that is left to do is to have some way to generate an infinite string of digits. Fractions have an infinite decimal representation.
For example, the fraction $\frac{37096}{33}$ has a decimal representation of $1124.\overline{12}$.
Converting the digit into inputs gives us the commands ↑↑→ (this picks up the key), ← and ↑→ infinitely. The last part will eventually reach the goal. The puzzle is considered solved if at any point the player reaches the goal.

Input

The first and only line contains two coprime integers a, b where $a \leq 2^{24}$ and $b \leq 2^{17}$, the numerator and denominator of the fraction $(\frac{a}{b})$.

Output

Output 1 if the fraction $\frac{a}{b}$ can solve the puzzle starting at S following the decimal representation of the fraction.
Output 0 otherwise.

Sample Input 1	**Sample Output 1**
37096 33	1

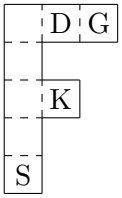
Sample Input 2	**Sample Output 2**
11241122 1	1

Sample Input 3	**Sample Output 3**
44 27	0

F2 Fractions playing games (Easy)

Time limit: 1s

Bob is coding a puzzle game. Bob is scared of making his game too hard, since the creator knows the solution to every puzzle. He has asked you to playtest one of the puzzles.



S is the starting location of the player. G is the goal the player needs to reach. However, the goal is locked behind a door D, for which the player will need to find a key K.
Being a programmer, you decide that the most efficient and straightforward way to solve this problem is to simulate random strings of inputs and check if the puzzle is solved.
Every input of the controller can be mapped to a digit in the following way.

- The ↑ input corresponds to 1 or 6. Move the player up 1 square. If the input makes the player cross a solid line, ignore the input.
- The → input corresponds to 2 or 7. Move the player right 1 square. If the input makes the player cross a solid line or a locked door, ignore the input.
- The ↓ input corresponds to 3 or 8. Move the player down 1 square. If the input makes the player cross a solid line, ignore the input.
- The ← input corresponds to 4 or 9. Move the player left 1 square. If the input makes the player cross a solid line, ignore the input.
- The pause/unpause input corresponds to 5 or 0. Switch the state of the game from unpause to pause or pause to unpause. Every movement input has no effect while the game is in the pause state.

All that is left to do is to have some way to generate an infinite string of digits. Fractions have an infinite decimal representation.
For example, the fraction $\frac{37096}{33}$ has a decimal representation of $1124.\overline{12}$.
Converting the digit into inputs gives us the commands ↑↑→ (this picks up the key), ← and ↑→ infinitely. The last part will eventually reach the goal. The puzzle is considered solved if at any point the player reaches the goal.

Input

The first and only line contains two coprime integers a, b where $a \leq 2^{24}$ and $b \leq 2^{17}$, the numerator and denominator of the fraction $(\frac{a}{b})$. The first 3 digits will be $1, 1, 2$. This guarantees that the key is picked. You don't have to implement the key verification in the easy version.

Output

Output 1 if the fraction $\frac{a}{b}$ can solve the puzzle starting at S following the decimal representation of the fraction.
Output 0 otherwise.

Sample Input 1	**Sample Output 1**
37096 33	1

Sample Input 2	**Sample Output 2**
11241122 1	1

Sample Input 3	**Sample Output 3**
44 27	0

G1 Giraffe excursion (Hard)

Time limit: 1s

Bob is coding a game with giraffes. He is having difficulty coding their movement. He wants them to move from the marshes of doom to the razor-sharp vine jungle. However, the shortest path would make them go through some nonsensical location, such as the cinema (c'mon, what would a giraffe do in a cinema?). He has assigned a plausibility factor to every road with the goal of preventing such implausible situations.

He could make them walk through the most enjoyable path until they reach their destination; however, Bob fears that his player base would get bored and stop playing his amazing game. He has decided to compromise. He will set a bound on the amount of time they have to go from location 1 to location 2. However, he wants to maximize the minimal plausibility factor of the travel.

For simplicity, the time it takes from one path to the other is proportional to the plausibility factor.

Given a weighted, undirected graph, find the route taken by the giraffes. The goal is to maximize the minimal plausibility factor along the path from location 1 to location 2 — i.e., the weakest (smallest-weight) road used on the route should be as large as possible. Among all routes that achieve this highest possible minimal road weight, choose the one with the smallest total time (the sum of plausibility factors along the path, proportional to time). If no route exists, output `impossible`.

Input

The input consists of:

- One line with two integers n and m, where n is the number of locations (nodes) and m is the number of roads (edges).
- m subsequent lines, each with three integers u, v, and w, describing an undirected road between locations u and v with plausibility factor w.

Nodes are numbered from 1 to n. There are no self-loops. Multiple roads between the same pair of locations may appear.

Output

Output a single integer: the total time of the shortest path from location 1 to location 2 among all routes whose smallest road (minimal plausibility factor on the route) is maximized. If no path exists, output `impossible`.

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

Sample Input 2	**Sample Output 2**
4 4 1 3 3 3 2 3 1 4 5 4 2 5	10

G2 giraffeexcursion easy

Time limit: 3s

Bob is coding a game with giraffes. He is having difficulty coding their movement. He wants them to move from the marshes of doom to the razor-sharp vine jungle. However, the shortest path would make them go through some nonsensical location, such as the cinema (c'mon, what would a giraffe do in a cinema?). He has assigned a plausibility factor to every road with the goal of preventing such implausible situations.
He could make them walk through the most enjoyable path until they reach their destination; however, Bob fears that his player base would get bored and stop playing his amazing game. He has decided to compromise. He will set a bound on the amount of time they have to go from location 1 to location 2. However, he wants to maximize the minimal plausibility factor of the travel.
For simplicity, the time it takes from one path to the other is proportional to the plausibility factor.
Given a weighted, undirected graph, find the route taken by the giraffes. The goal is to maximize the minimal plausibility factor along the path from location 1 to location 2 — i.e., the weakest (smallest-weight) road used on the route should be as large as possible. Among all routes that achieve this highest possible minimal road weight, choose the one with the smallest total time (the sum of plausibility factors along the path, proportional to time). If no route exists, output `impossible`.

Input

The input consists of:

- One line with two integers n and m, where n is the number of locations (nodes) and m is the number of roads (edges).
- m subsequent lines, each with three integers u, v, and w, describing an undirected road between locations u and v with plausibility factor w.

Nodes are numbered from 1 to n. There are no self-loops. Multiple roads between the same pair of locations may appear.

Output

Output a single integer: the total time of the shortest path from location 1 to location 2 among all routes whose smallest road (minimal plausibility factor on the route) is maximized. If no path exists, output `impossible`.

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

Sample Input 2	**Sample Output 2**
4 4 1 3 3 3 2 3 1 4 5 4 2 5	10