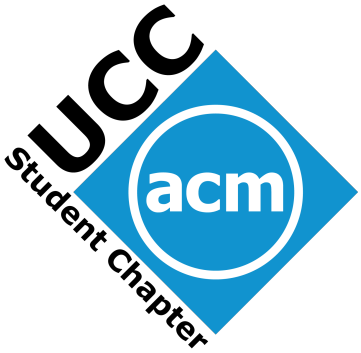
Irish Collegiate Programming Competition 2021

January 23, 2023



Contributors

The ACM Student Chapter committee would like to thank the following people.

For leading the question writing:

- Dr Sabin Tabrica
- Marcu Bogdan Stefan
- Colm Hickey
- Andrew Nash

For reviewing and editing the questions:

- Brian McCarthy
- Jakub Piatek

1 Planned Production Line Problems

The manager of an electric vehicle manufacturing plant has become too preoccupied with ambitions of space colonisation, to have time to monitor their cutting edge AI robotic car assembly line.

You have been tasked with implementing a system to track and report on the status of the different robotic arms on the line.

The arms are arranged sequentially along a conveyer belt, and are numbered from $0, 1, \ldots n$. Each arm has an Operational Percentage Indicator (OPI), a value from 0 to 100 that indicates to what extent it is working at full capacity.

Your system has to be able to handle the following queries, given an initial sequence of OPIs.

When an arm fails, we no longer consider its OPI when examining the OPIs of the arms along the line. Further, after a failure, the indices of all arms down-line from the failure are decreased by one.

An engineer has speculated that failures are more likely between adjacent machines, possibly due to climactic conditions.

- Set the OPI of arm x to value O
- Get the mean, and minimum OPI of all arms in range (a, b)
- Handle the failure of arm x
- An engineer has speculated that failures are more likely between adjacent machines, possibly due to climactic conditions. We occasionally want to check if the (operational) arm at index i is adjacent to a failed arm

Example

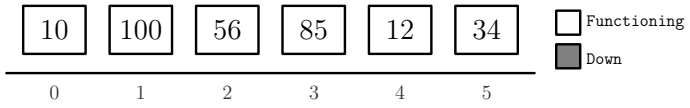


Figure 1: Initial configuration of the production line, showing 6 arms with different OPIs.

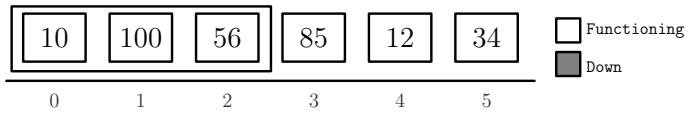


Figure 2: Querying range (0,2).

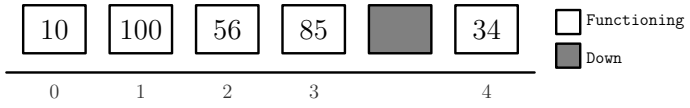


Figure 3: Arm 4 reports failure.

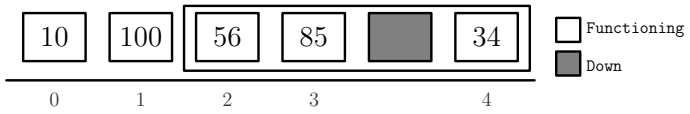


Figure 4: Querying range (2,4).

Input/Output

The first line of input is a single integer N, the length of the assembly line. This is followed by N integers between 0 and 100, the OPIs of each arm on the line.

This is followed by a single integer Q, and then Q lines where

- If the first character of the line is F, it will be followed by a single integer - the index of the arm that has reported failure.
- If it is a Q, then you need to perform a query between the two indices that follow. You should print the result of the query to the standard output, as two space separated integers (average followed by minimum of the range).
- If it is a C, we need to check if the arm at the given index is adjacent to a failed arm. If so, print the string YES to the standard output, otherwise print NO

Examples

Sample Input 1

```
6
10 100 56 85 12 34
5
Q 0 2
F 4
Q 2 4
C 0
C 3
```

Sample Output 1

```
166 10
175 34
NO
YES
```

This example corresponds to the input illustrated above.

Constraints

- $2 \leq N \leq 2\,000\,000$.
- $1 \leq OPI_i \leq 100$.
- $1 \leq Q \leq 5000$

2 Carefully Crafting Customs Clearance

In this new and uncertain post Brexit world, there is only one certainty - taxes.

As a person who regularly dispatches packages to international clients, this situation is extremely troubling to you. A grim possibility has occurred to you, that you want to be sure to be prepared for.

Consider the following hypothetical situation:

- All packages have to have a customs clearance slip attached. This is valid for the N days (until 23:59:59 on the Nth day, if it was stamped on day 0) after it was most recently stamped by a customs official.
- Purchasing initial export clearance from your own country costs C euros per day of validity,i.e the total cost for the slip is $C \times N$
- Some countries have the authority to stamp and re-approve a clearance, allowing the package to be valid for N days after this point. Note that re-approval does not affect the initial cost of the slip.
- Some countries have very limited international approval abilities - these countries can approve the package for $\frac{Z}{2}$ (ignoring any decimal information) days, if this is bigger than the outstanding validity on the package. Note that Z corresponds to the lowest number of outstanding days of validity this package had at any point in time. That can potentially be the number of outstanding days on the validity of the package when it arrived at this country.
- A package can only pass through or arrive in a country if it is valid. If it is invalid, the package will be incinerated.

You are given C, and a series of countries through which a package must pass, along with a destination, and the number of days that the package will take to travel between each.

Find the minimal value of N that you have to pay for to ensure the package reaches its destination. A package starts fully approved at its source.

Input/Output

The first line contains integers C, M, the initial purchase cost per day of validity, and number of countries through which the package must pass (including the source and destination).

$M - 1$ lines follow. Line i contain an integer x_i, the number of days it takes the package to ship from country $i - 1$ to country i. This is followed by a space, then a single character, F or P. F denotes that country i has the power to issue full re-approval, P denotes partial approval authority. The destination's approval status is irrelevant, as it only has to assess existing approvals, but one is specified anyway for consistency.

Output a single integer, the cost to approve this package

Examples

Sample Input 1

```
6 3
4 P
2 F
```

Sample Output 1

```
36
```

Sample Input 2

```
1 3
2 P
5 F
```

Sample Output 2

```
7
```

Constraints

- $1 < C < 10000$
- $2 < M < 550000$
- $1 < x_i < 1000000$

3 Big Bad Binary Warehouse

You have successfully created a web crawling robot that has managed to buy all the available consumer electronics online in Europe. With the market cornered, vast riches await. Alas, you have encountered a problem! While you raised enough capital to buy the devices, you forgot to set up a warehouse to store them. As a substitute, you have purchased a large cattle shed, which you then re purposed.

With the space secured, you now just have to implement an inventory management system. There is a very large number of items that you need to stash in the shed, but you have to store very little information about each item - just an integer id that represents the type of product that it is. You can pick arbitrary ids for different items, but the condition must hold that different types of items have different ids. Further, ids are represented in the most straightforward binary form - no two's complement, special encoding, ECC, sign bits etc can be used.

Further, you are running on an old and temperamental server. If a long series of consecutive 1s or 0s is written to disk, that sector of the disk will fail. Your purchasing actions have left a great number people very angry and upset, and now nobody seems willing to sell you a new enterprise grade server. As a consequence you have to work with the bad hardware, and implement your own wear leveling strategy on the server's operating system to prevent disk failure. (A wear levelling strategy is a process whereby data is distributed on a disk in such a way as to maximise the longevity of the hardware involved).

The solution here is clearly to make sure that the binary representations of the ids of your stock items contain no long series of 1s or 0s. You know that you are to expect huge deliveries of N types of items, so we need to choose N distinct ids that have no long 0 or 1 subsequences.

After testing a spare server to destruction, you discover that a block of data is guaranteed to not cause disk failure if any arbitrary continuous subsequence of the block's binary data has no more than 3 more 0s than 1s, or vice versa.

As a final constraint, in order to make your database consistent, all id must be integers represented by the same number of bits.

What is the minimum number of bits needed to represent N integer ids safely on our server, under the constraints above?

0 0 1 1 0 1

Figure 5: An example of a valid binary id that can be safely written to the server.

0 1 1 1 0 1 1 0 1

Figure 6: An example of an invalid binary id that cannot be safely written to the server. In the boxed subsequence, there appear five 1s and one 0. There is a difference of 4, which is greater than 3 and thus damage is caused

Input/Output

There is only one line of input, containing a single integer N.

Output a single integer, the necessary number of bits

Examples

Sample Input 1

```
3
```

Sample Output 1

```
2
```

Sample Input 1

```
100
```

Sample Output 1

```
8
```

Explanation of Sample Input

Sample Input 1 There are four ways to represent 3 ids with 2 bits of data

```
00,01,10
00,01,11
00,11,10
11,01,10
```

Any of these could be used, as they are all valid. There are only two possible ids that can be represented with 1 bit: 0 and 1. Thus, 3 bits is the minimal solution.

Constraints

- $1 \leq N \leq 18\,446\,744\,073\,709\,551\,615$.

4 N-Dimesnional Chess Check Checker

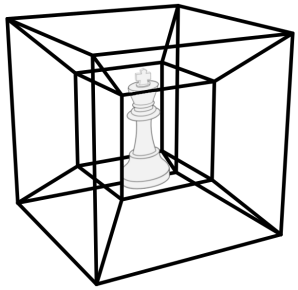


Figure 7: There is no way to graphically illustrate an N dimensional space, but enjoy the illustration

The phrase "3-Dimensional Chess" is used to refer (informally) to extremely complex, dynamic and highly sensitive systems. The idea of this is that 3-Dimensional Chess would have much more complexity and require more skill than the more traditional game played in 2 dimensions.

As a scientist, you know however, that one doesn't need to constrain oneself to systems of any specific number of dimensions. Enter N-Dimensional Chess - a generalised version of the game of chess that does not constrain us to any specific one of the countably infinite dimensions we could be playing in.

You have managed to attract, bribe, and coerce your colleagues into playing this game, but have encountered a problem. Not everyone is clear on the rules, and in particular, many people do not know if when they are in a check or a checkmate situation.

Given a game of N-Dimensional chess, with a specific value of N, and the positions of all pieces on the board, state if there is a check situation (no need to check for checkmate).

Obviously, the rules of movement from normal chess have to be modified slightly to work here.

- **Pawns** Pawns can move one unit in any of the N axis directions, in a positive manner. Pawns can take any piece further away from the origin than it, that is on a diagonal of 1 unit magnitude in each axis direction. (See Bishop for clarification on diagonal movement)
- **Rooks** Rooks can move arbitrary units in any of the N axis directions. They can take any piece at the end of such a path.
- **Knights** Knights can move two units in any axis direction, then one in another different axis direction. They can jump over any pieces on this path, something other pieces can't do.
- **Bishops** Bishops can move diagonally through the space. In practice, this is defined as the **magnitude** of the change in each and every axis direction is the same. Bishops can take any piece at the end of such a path.
- **Queens** Queens can make all of the same moves as the above (except Knights), without any of the constraints.
- **Kings** Kings can move 1 unit in any axis direction, as well as make a diagonal move of 1 unit (magnitude of change in every axis direction is 1), and take any piece that isn't a king on this path. A king cannot move into a position which would put it in Check.

Input/Output

The first line contains an integer G, the number of games to adjudicate. Each of these G cases starts with two integers, N, P - the dimension of the specific game to follow, and the number of pieces in play.

P lines follow.

Each line starts with one of W or B, followed immediately by P,R,k (knight),B,Q or K (King) - the colour and type of piece in question. A space, and N space separated integers follow - the coordinates of this piece on each of the N hyperboard's axes.

For each case, if you detect a check on white king, output "WHITE CHECK FROM B*", where * is the type of piece causing check. Similarly for a black check, "BLACK CHECK FROM W*". If there are multiple pieces causing a check, use the one that appears lowest in the bullet pointed list above. If no check is detected, output "NO CHECK"

Examples

Sample Input 1

```
2
1
3
BK 3 3
BP 2 3
Wk 2 1
```

Sample Output 1

```
BLACK CHECK FROM Wk
```

Sample Input 2

```
4
1
2
WK 1 2 3 1
BB 4 -1 0 4
```

Sample Output 2

```
WHITE CHECK FROM BB
```

Sample Input 3

```
4
2
3
WP 3 0 1 3
WK 1 2 3 1
BB 4 -1 0 4
4
WK 1 2 3 1
BQ 4 -1 0 4
BR 1 -17 3 1
BP 0 3 3  1
```

Sample Output 3

```
NO CHECK
WHITE CHECK FROM BR
```

Constraints

- $2 < N < 5000$
- $2 < P < 1000$

5 N-Dimesnional Baskethyperball

So, the N-Dimensional chess with its contrived rules didn't work out, but never fear, you have a devised a cunning plan!

The most basic rules of basketball are (ostensibly) easier to understand than those of chess. One throws the ball into the hoop, and is awarded points according to the distance of the shot.

This is much easier to convert into an N-Dimensional game! Note that a hypersphere is a generalisation of a sphere - it is the set of points within its radius of its centre, in all N dimensions

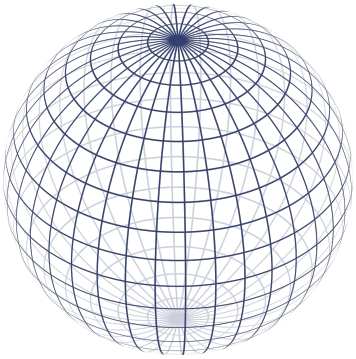


Figure 8: There is no way to graphically illustrate an N dimensional object, but you can imagine that this is a 3D cross section of one. Credit: Geek3, CC BY-SA 3.0

This time, people have joined the game of their own volition (see, already a success!). All you need to do is implement an automatic electronic scoreboard. The hyperball (a generalisation of a ball in N dimensions), contains an NGPS (N-Geometric Positioning System) chip, that reports the coordinates of the centre of the ball over WiFi. The single hyperhoop (a hyperball, or circle of 1 dimension less than the playing hyperball) is placed at a fixed point in space. I.e, when the ball is a sphere, the hoop is a circle with given radius at a fixed height above the ground, when the ball is a 4-D hypersphere, the hoop is a regular 3-D sphere with a certain radius, at a given "height" or first dimension. A ball is only in the hyperhoop if its height is exactly that of the hoop, and lies inside the hypershpere it forms in the other dimensions.

For simplicity, you can assume the playing hyperball has negligible radius.

While the game takes place in N dimensions, we do want to maintain some notion of a level planar court. For this reason, we denote the first of these N dimensions as "height" above the surface of the court. Players are constrained to remain at the same "height" for the whole game, and the hyperhoop is positioned at a fixed height. For example, in plain 'ol 3D basketsphere, the hoop is exactly 3 metres above the ground, and players tend not to get taller or shorter mid-game.

Given a radius and height for the hyperhoop, and a set of N coordinates that give the position of the hyperball, state whether the ball is inside the hoop at that moment.

Input/Output

The first line is N, the dimensionality of the space in which the game is played..

The next line contains a floating point numbers R, the radius of the hyperhoop. There follow N floats, the N coordinates of the centre of he hoop. The first of these corresponds to its height.

There then comes an integer, M, the number of shots to check. Each of the following M lines each contain N floats, the coordinates of the centre of the ball for each shot.

For each of the M cases, output "HIT", if the hyperball is in the hyperhoop for that case, and "MISS" otherwise.

Examples

Sample Input 1

```
3
1.4 2 0 0
6
1 0 0
3 0 0
2 0 1
2 1 1
2 3 1
2 3 4
```

Sample Output 1

```
MISS
MISS
HIT
HIT
MISS
MISS
```

Sample Input 2

```
4
10.4 2 1 0 1
6
1 0 0 -3
3 0 0 3
2 0 0.5 0.5
2 1 1 0.75
2 30 1 0.5
2 3 4 1
```

Sample Output 2

```
MISS
MISS
HIT
HIT
MISS
HIT
```

Constraints

- $2 < N < 5000$
- $2 < M < 1000$

6 Massive Watermain Misspending

As a project manager assigned to building a new town (in a very ordered society), you have paid a contractor to lay piping for the town's water network. The town has N main water exchanges. Water will enter one of these from the national pipeline, and return to the national pipeline from a different exchange. These are known as the source and drain. Alas! The contractor did a terrible job - they laid far more pipes than you needed, and never connected to the national pipeline.

We need to finish the job ourselves - select the pipes that we want to use for our network, and connect our source and drain to the national pipeline, while maximising the flow in out network.

To comply with regulations, every exchange must be connected in a specific order, as a loop. Note that if an exchange is designated as a drain, the previous exchange must be designated as a source, and the two exchanges do not have to be connected.

Pipes

- Pipes between exchanges are connected by junctions which have unique integer numbers greater than N
- There are no other guarantees about the values, or order of the numbers of the junctions, other than that they are all unique.
- It is guaranteed that a pipe will lie on a direct path (one that does not have to pass through another exchange), between exactly two exchanges.

Flow

- The flow between two exchanges is the minimum capacity of the pipes that the flow is through.
- The flow of the network is the minimum of all the flows between any exchanges.

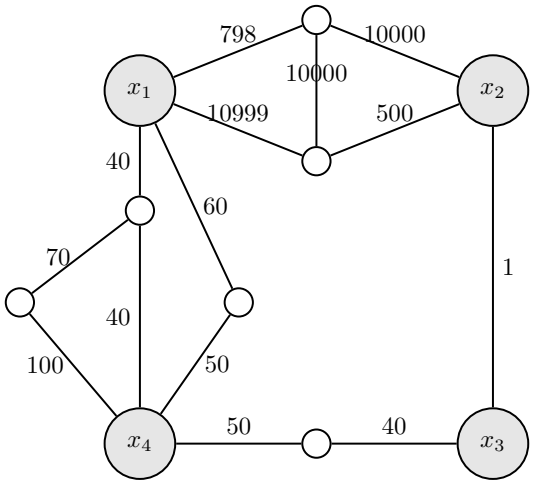


Figure 9: An example network of pipes. Grey nodes denote the 4 main exchanges

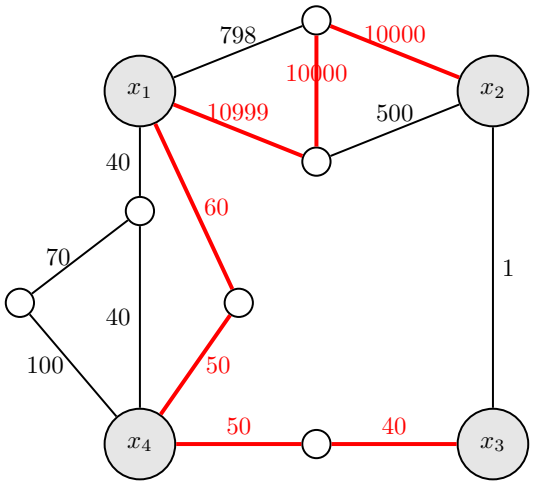


Figure 10: The optimal solution for the above network. Nodes x_2 and x_3 are designated source and drain respectively. The overall capacity of this network is 40, which is minimal.

Input/Output

The first line of input is N, the number of exchanges.

Following this, we repeat the following procedure from $i = 1, \ldots N$, to read the network of pipes between exchanges i and $i + 1$

Read a integers j, p denoting the number of junctions (points where two or more pipes meet) and pipes between exchanges i and $i + 1$.

There then follow p lines, each containing 3 integers, a, b and w - this represents a pipe with capacity w running from junction/exchange a to junction/exchange b.

Output 3 integers, the source, the drain, and the capacity of the completed network.

Sample Input 1

```
4
2 5
1 5 798
5 6 10000
1 6 10000
5 2 10000
6 2 500
0 1
2 3 1
1 2
3 7 40
7 4 50
3 6
4 8 100
4 10 40
4 9 50
9 1 60
8 10 70
10 1 40
```

Sample Output 1

```
2 3 40
```

This example corresponds to the input presented above.

Constraint

- $1 \leq N \leq 4\,000$
- $1 \leq \text{junctions} \leq 500$
- $1 \leq \text{pipes} \leq 125\,000$

7 Busy Marina

The marina in a small village is very badly organized with boats anchored randomly in either vertical or horizontal positions, relative to the land. Aerial photography has given a clear image of how the boats have been chaotically placed in the marina.

John, the marina manager, has reduced the photo to a matrix with N rows and M columns in which the boats are represented by consecutive sequences of 1's and the water by ares of 0's. A boat is represented by at least 2 consecutive 1's, depending on its size - smaller boats are represent fewer 1's and longer boats with many consecutive of 1's.

The boats are all placed such that they do not make contact, or collide, i.e. a vertical boat cannot touch a horizontal boat.

Note that there may be in the map some noise, created by John's image compression, that may result in some isolated 1's (surrounded by 0's on all four sides) which does not represent a boat. Given such an N by M matrix, determine how many boats are anchored in the marina

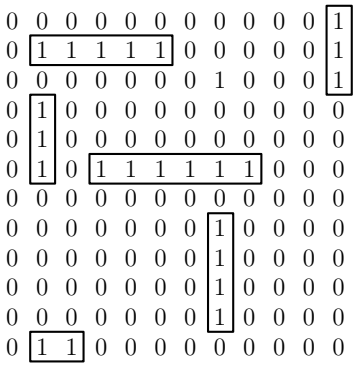


Figure 11: An example 12x12 marina, with the six boats indicated

Input/Output

The first two lines contain two integers, N and M, corresponding to the height and width of the matrix

The next N lines contain M space separated 0's and 1's.

Output the number of boats in the matrix

Examples

Sample Input 1

```
12 12
0 0 0 0 0 0 0 0 0 0 0 1
0 1 1 1 1 1 0 0 0 0 0 1
0 0 0 0 0 0 0 1 0 0 0 1
0 1 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0 0 0 0 0
0 1 0 1 1 1 1 1 1 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0
0 0 0 0 0 0 0 1 0 0 0 0
0 1 1 0 0 0 0 0 0 0 0 0
```

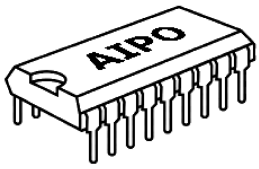
Sample Output 1

```
6
```

Constraints

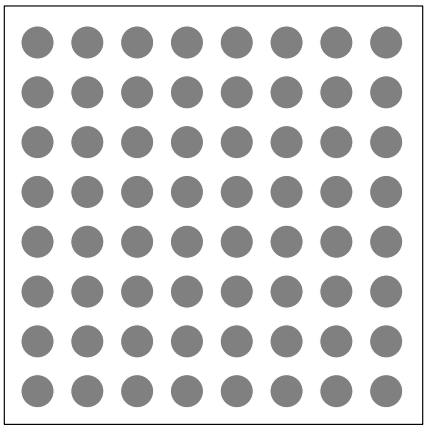
- $0 \leq N, M \leq 1000$

8 Largest LGA Chip Placement



You are a penetration testing specialist that has been tasked with exfiltrating data from a client's data centre. After cloning the badge of a security guard, you have succeeded in gaining access to the central server room.

Unfortunately there are no peripherals or ports that allow you to interact with the servers. Out of the corner of your eye, you spot a server that has its motherboard with a land grid array (LGA) socket exposed. You know that you can exploit this to achieve your goal.



(a) An example of a Land Grid Array socket

```
0V 5V 5V 5V 0V 0V 0V 5V
0V 0V 0V 5V 0V 0V 5V 0V
5V 5V 5V 5V 5V 0V 0V 0V
0V 0V 0V 5V 0V 5V 5V 0V
0V 0V 5V 0V 0V 5V 5V 0V
5V 5V 0V 5V 5V 5V 5V 0V
5V 0V 5V 0V 5V 5V 5V 0V
5V 0V 0V 5V 0V 5V 5V 0V
```

(b) An example power state of an 8x8 LGA socket

Using your portable volt-meter you have established which of the pads are powered and which ones are drains (have no power).

You have to find the area of the biggest chip that you can place on the board without causing a short circuit. A short circuit is caused by your chip if the amount of voltage pads and non-voltage pads that it covers is not equal. You have access to chips of any rectangular shape.

Input/Output

The first two lines contain two integers, N and M, corresponding to the height and width of the LGA socket.

The following N lines contain M integers, the voltage level V_{ij} at that position on the socket

Output the size of the maximum area containing an equal number of 5's and 0's. If there is nowhere to place the chip without causing a short circuit, you should output "No possible chip"

Examples

Sample Input 1

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

Sample Output 1

```
8
```

Sample Input 2

```
2 4
0 0 5 5
0 5 5 5
```

Sample Output 2

```
6
```

Sample Input 3

```
1 1
0
```

Sample Output 3

```
No possible chip
```

Constraints

- $0 \leq N, C \leq 200$.
- $0 \leq V_{ij} \leq 5$

9 Some Curious Primes

Primus has just learned to convert a number between bases through consecutive divisions. As Primus is very fond of prime numbers, they started to decompose the regular base-10 primes in all the other bases from 2 to 10. For some primes, Primus observed that the sum of their digits may also be prime across all bases from 2 to 10 and they called them "curious primes".

Given a number N then check if it is curious prime or not.
Output "YES", if N is a curious prime, and "NOT" otherwise.

Input/Output

A single integer, N

Examples

Sample Input 1

```
13
```

Sample Output 1

```
NOT
```

Sample Input 2

```
131
```

Sample Output 2

```
YES
```

Explanation of Sample Input 1

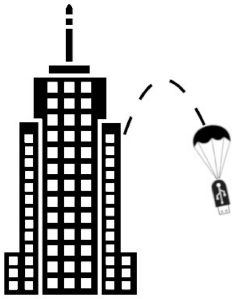
$$(13)_{10} = (1101)_2 = (111)_3 = (31)_4 = (23)_5 = (21)_6 = (16)_7 = (15)_8 = (14)_9$$

And, $1 + 3 = 4$,$3 + 1 = 4$ and $1 + 5 = 6$ (for bases 10,4,8) are all non-prime, and so violate the conditions for curious primes.

Constraints

- $0 \leq N, C \leq 2\,000\,000\,000$.

10 Parachuting USB



You are a penetration testing specialist with a mission of exfiltrating data from a large and secure office building. Together with your team you have already planned on how to infiltrate the building and copy the sensitive data onto a USB drive. There is strong security at the door, so it seems that the only way of extracting the USB drive from the building undetected is to throw it out of a window attached to a small parachute, to be retrieved from the street below. The sensitive data is kept on

the ground floor, which is where you have to perform the copying. There is a great risk of being caught, as you ascend the building, so you want to extract the USB as quickly as possible. For this reason, you need to know the lowest possible floor from which you can safely extract. The day before

the mission you and your team want to determine which is the lowest floor from which you can drop the drive without it getting destroyed. If the level is too small the parachute will not have enough time to open, and the drive will hit the floor at unacceptably high speed. If the level is too high, you run unnecessary risk of being caught. Luckily, you find a similar building for which the owner allows you to do your experiments, and determine the exact optimal floor. Unfortunately, they require you to pay each time you drop the USB drive, as it requires you to be walking through their offices each time, causing some disturbance.

Knowing that you have a certain amount of test drives that you can throw, what is the minimum amount of drops you need to perform to **guarantee, for any possible safe level** that you find the lowest safe level the USB drive can be dropped from.

- A USB drive that survives a test fall can be used again. Test falls that are successful do not effect the longevity of the USB
- A broken USB drive must be discarded, as it it irreparably broken
- The effect of a fall is the same for all USB drives
- If an USB drive breaks when dropped from a floor, it would have broken if it had been dropped from any lower floor
- If an USB drive survives a fall from a floor then it would survive a longer fall, form any floor above that one
- It is neither guaranteed nor ruled out the the first floor is safe or breaks the drive
- It is neither guaranteed nor ruled out the the last floor is safe or breaks the drive

What is the least number of USB drive drops that is guaranteed to work in all cases?

Input/Output

Two lines, containing integers N, the number of floors of the target office building and M the number of test USB drives at our disposal, respectively.

Output a single integer, the minimum amount of test drops necessary to know with certainty the lowest safe level from which the USB can be ejected.

Examples

Sample Input 1

```
4
1
```

Sample Output 1

```
4
```

Sample Input 2

```
36
2
```

Sample Output 2

```
8
```

Constraints

- $0 \leq N \leq 10\,000$.
- $0 \leq U \leq 10$