Nordic Collegiate Programming Contest

NCPC 2023

October 7, 2023

Problems

A Aperiodic Appointments
B Berry Battle 2
C Converting Romans
D Die Hard
E Electronic Components
F Factor-Full Tree
G Groups of Strangers
H Heroes of Velmar
I Intertwined
J Jamboree
K Karl Coder

Do not open before the contest has started.

Advice, hints, and general information

- The problems are **not** sorted by difficulty.
- Your solution programs must read input from *standard input* (e.g. `System.in` in Java or `cin` in C++) and write output to *standard output* (e.g. `System.out` in Java or `cout` in C++). For further details and examples, please refer to the documentation in the help pages for your favorite language on Kattis.
- For information about which compiler flags and versions are used, please refer to the documentation in the help pages for your favorite language on Kattis.
- Your submissions will be run multiple times, on several different inputs. If your submission is incorrect, the error message you get will be the error exhibited on the first input on which you failed. E.g., if your instance is prone to crash but also incorrect, your submission may be judged as either "Wrong Answer" or "Run Time Error", depending on which is discovered first. The inputs for a problem will always be tested in the same order.
- If you think some problem is ambiguous or underspecified, you may ask the judges for a clarification request through the Kattis system. The most likely response is "No comment, read problem statement", indicating that the answer can be deduced by carefully reading the problem statement or by checking the sample test cases given in the problem, or that the answer to the question is simply irrelevant to solving the problem.
- In general we are lenient with small formatting errors in the output, in particular whitespace errors within reason, and upper/lower case errors are often (but not always) ignored. But not printing any spaces at all (e.g. missing the space in the string "`1 2`" so that it becomes "`12`") is typically not accepted. The safest way to get accepted is to follow the output format exactly.
- For problems with floating point output, we only require that your output is correct up to some error tolerance. For example, if the problem requires the output to be within either absolute or relative error of 10^{-4}, this means that
  - If the correct answer is 0.05, any answer between 0.0499 and $.0501$ will be accepted.
  - If the correct answer is 500, any answer between 499.95 and 500.05 will be accepted.

  Any reasonable format for floating point numbers is acceptable. For instance, "17.000000", "0.17e2", and "17" are all acceptable ways of formatting the number 17. For the definition of reasonable, please use your common sense.

Problem A
Aperiodic Appointments

Nick has always struggled with maintaining habits. The problem is that he just can't stop maintaining them. If Nick does something K times in a row, he has to keep doing it forever.

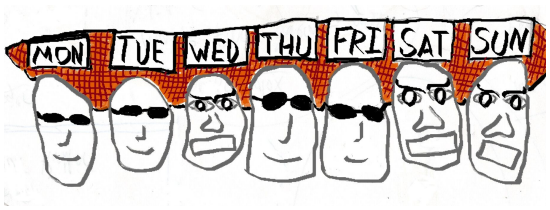
Luckily, he has started visiting Dr Patternson, an expert in PBT (Pattern Breaking Therapy). The principle of PBT is simple: Nick will visit Dr Patternson every day, and if he has done the same thing K times in a row on a specific visit, the doctor will charge him money. This will motivate Nick to not continue this habit.

PBT has worked out great for Nick, as he has now successfully quit all his habits. Except for one, the habit of visiting Dr Patternson. The frequent visits are starting to take a toll on Nick's economy, so your task is to calculate how many times he has to pay the doctor for the next N days.

Formally, let $s = s_1 s_2 s_3 \ldots s_N$ be a string consisting of zeroes and ones. A one means that Nick has to pay the doctor on the ith day. This string is generated one character at a time, in the following way:

1. $s_i = 0$ if $i \leq K$.
2. If $i > K$, then $s_i = 1$ if the previous characters contains a pattern that repeats K times. More specifically, let $s' = s_1 s_2 \ldots s_{i-1}$. If there is a nonempty string t such that the last $|t| \cdot K$ characters of s' can be written as $t + t + \cdots + t$, then $s_i = 1$. Otherwise $s_i = 0$.

You are given the numbers N and K, and your task is to calculate the number of ones in the string s.



The picture represents Sample 1. An angry face means that Nick had to pay on the corresponding day.

Input

The input consists of one line with the integers N and K ($1 \leq N \leq 10^9$, $2 \leq K \leq 10^9$).

Output

Print one integer, the number of ones in the string s.

Sample Input 1	Sample Output 1
7 2	3

Sample Input 2	Sample Output 2
99 5	19

This page is intentionally left blank.

Problem B
Berry Battle 2

Picture by MAKY.OREL, public domain

Erik and his grandpa often go to the forest to pick blueberries. Grandpa always finds the most berries, and even though it is not a competition, Erik has had enough of this. He has observed that grandpa uses a simple greedy strategy to pick as many berries as possible. With this information, Erik will try to finally pick at least as many berries as grandpa.

The blueberry shrubs can be represented as a string $s = s_1 s_2 \ldots s_N$ of length $N = 10^5$, consisting of characters `.` and `b`. If $s_i = \texttt{b}$, then there is a berry at position i, otherwise there is no berry there. There will be exactly $N/2$ berries to start with, and they are distributed uniformly at random.

The berry picking takes place in turns. In one move, a person can choose a position i ($1 \leq i \leq N - 3$), and pick all the berries at positions i, $i + 1$, $i + 2$, and $i + 3$. Grandpa makes the first move, then Erik makes a move, then grandpa, and so on. Grandpa will always greedily make a move that picks as many berries as possible. Among all such moves, he will pick one uniformly at random.

You are given the string s, and your task is to write a program that decides what moves Erik should make in order to pick at least as many berries as grandpa.

Interaction

The first line of input contains the number N, the length of the string s. This number will always be equal to 10^5, except for in the sample. Your program does not have to solve the sample to get accepted.

The second line contains the string s of length N. This string has exactly $N/2$ characters `b`, and their positions are generated uniformly at random.

Then, your program should start interacting with the grader. When it is grandpa's turn, you should read a single integer i ($1 \leq i \leq N - 3$), meaning that grandpa picks the berries at positions i, $i + 1$, $i + 2$, and $i + 3$. When it is your turn, you should instead print a number i, meaning that you pick the berries at positions i, $i + 1$, $i + 2$, and $i + 3$.

If it is your turn and there are no more berries, your program should terminate without printing anything more. If it is grandpa's turn and there are no berries, then the grader will not make any more moves, and your program should also terminate without printing anything more. If you picked at least as many berries as grandpa, you pass the test case.

Each turn, grandpa greedily makes a move that picks as many berries as possible, and among all such moves, he chooses one uniformly at random.

The strings s were generated beforehand, so they are the same across different submissions. However, grandpa's behaviour is randomized every submission, so it is possible to get different results if you submit the same code twice.

There will be 100 test cases. It is guaranteed that there exists a solution that passes this many cases with very high probability.

Example

In the sample, Erik and grandpa get 5 berries each, which means that Erik achieved his goal. At the very start of the game, the maximum number of berries possible to pick in one move is 3. There are a 4 possible choices of i that gets 3 berries: $1, 2, 3$ and 6. Grandpa chooses one of these uniformly at random which ends up being $i = 2$. After that, Erik chooses $i = 6$ which gives him three berries as well. In the last two turns, both grandpa and Erik pick two berries.

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

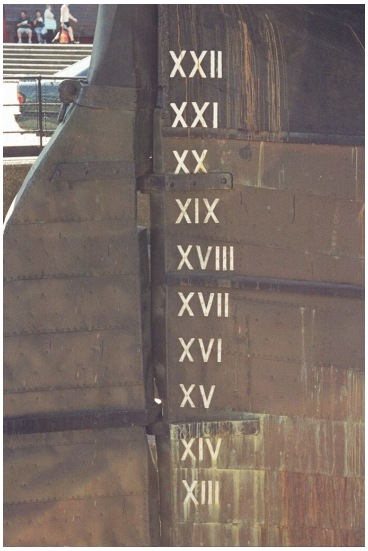
```
20
.bbb.b.bb...b.b...bb
2
```

```
6
```

```
13
```

```
17
```

Problem C
Converting Romans



Roman numerals on stern of the ship Cutty Sark showing draught in feet. The numbers range from 13 to 22, from bottom to top. (CC BY-SA 3.0)

The Roman numerals are a numeral system that originated in ancient Rome and was widely used throughout Europe well into the Late Middle Ages. It differs from the Arabic system that we mostly use today in that numbers are written with combinations of letters from the Latin alphabet, where each letter is assigned a fixed integer value. Over the years many different letters were used, which was a cause of frequent confusion. But "modern" style uses only these seven:

I	V	X	L	C	D	M
1	5	10	50	100	500	1000

A very common misunderstanding of the Roman numerals is that they use a universal subtractive syntax. In a subtractive syntax, a lower digit written before a larger digit will be subtracted from the larger digit. 4 can be written as IIII and IV $(5 - 1)$. Romans themselves only really used the subtractive syntax for smaller numbers. For larger numbers, it was largely avoided, to give more clarity. For example, 499 can be written LDVLIV, XDIX, VDIV or ID, while older sources most likely would use CDXCIX. Ain't Roman numbers fun?

You are given n numbers written in the Roman numeral system, and your task is to convert them to the regular Arabic number system.

For the purposes of this problem we will subtract any digit written to the left of a larger digit, even if they are not directly adjacent.

Input

The inputs starts with an integer $0 < n \leq 10^3$. Then follow n lines, each containing up to $3 \cdot 10^5$ Roman digits. The total number of Roman digits in the input is at most $3 \cdot 10^5$.

Output

Output the corresponding Arabic numbers, one number per line.

Sample Input 1	Sample Output 1
5 CDXCIX ID IV IIIIV IIIIIIV	499 499 4 1 -1

This page is intentionally left blank.

Problem D
Die Hard

Miwin's dice by Dr.M.Winkelmann, public domain

John and Hans are playing a game involving 3 dice. Even though they are all 6-sided, they are not guaranteed to be identical.

First John picks one of the dice and then Hans picks one of the remaining two. Then they both roll their chosen die. If they roll the same number, they both re-roll their die. Otherwise the winner is the one who rolled the highest number.
In case neither John or Hans can win with their chosen dice, they do not bother to re-roll the dice indefinitely and no winner is declared.

Can you help John pick a die that guarantees that he wins with a probability of at least $\frac{1}{2}$?

Input

The input consists of three lines. Line i contains 6 positive integers x_j ($1 \leq x_j \leq 1000$), describing the sides of the i'th die.

Output

Output the smallest $i \in \{1, 2, 3\}$, such that John can pick die i and be guaranteed to win with probability at least $\frac{1}{2}$. If no such die exists, output "No dice".

Sample Input 1

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

Sample Output 1

```
1
```

Sample Input 2

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

Sample Output 2

```
3
```

Sample Input 3

```
2 2 4 4 9 9
1 1 6 6 8 8
7 7 5 5 3 3
```

Sample Output 3

```
No dice
```

Sample Input 4

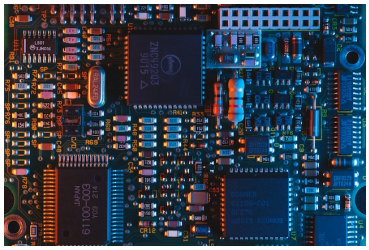
```
1 1 1 1 1 1
2 2 2 2 2 2
2 2 2 2 2 2
```

Sample Output 4

```
No dice
```

This page is intentionally left blank.

Problem E
Electronic Components



Picture by Umberto, Unsplash Licence

Sara is doing her summer internship at NCPC (Never Crashing Personal Computers). One day, a rare creature appeared in the office: an algorithmic problem!

The company has a machine that places electronic components on circuit boards. Normally, it would do this one component at a time. But recently the machine has received an update which allows it to place two different components simultaneously. The bottleneck then becomes the component with greater placement time. Now it is far from obvious what strategy the machine should use in order to minimize the total placement time. Sara decides to write an algorithm to determine this strategy.

You have N different types of electronic components. There are f_i copies of the ith type, and the components of this type have a placement time of t_i nanoseconds. The goal is to place all of the components using a sequence of moves. In one move, the machine can take two components of type i and j, where $i \neq j$, and place both of them simultaneously. This takes $\max(t_i, t_j)$ nanoseconds. The machine can also place a single component of type i in one move, which takes t_i nanoseconds.

Calculate the minimum possible time to place all components.

Input

The first line of input contains the integer N ($1 \leq N \leq 1000$).

The following N lines each contain two integers f_i and t_i ($1 \leq f_i \leq 10^4$, $1 \leq t_i \leq 10^9$).

Output

Print one integer, the minimum time to place all components.

Sample Input 1	Sample Output 1
3 2 7 2 1 3 10	31

Sample Input 2	Sample Output 2
3 2 10 2 11 2 12	35

Sample Input 3	Sample Output 3
4 2 11 7 10 3 5 1 1	72

Problem F
Factor-Full Tree

Aivar is very good at number theory. In fact, it is the only thing he is good at, but this doesn't stop him from achieving great things. However, if Aivar wants to solve any problem in life, he first needs to convert it to number theory.

For example, consider a rooted tree with N vertices. In order to deal with such structures, Aivar first constructs a *divisor labelling* of the tree. A divisor labelling is a way to label each vertex v with a positive integer x_v so that v is an ancestor of u if and only if x_v divides x_u.

After constructing such a labelling, Aivar can simply forget about the tree and just think about the list of numbers $x_1, x_2, \ldots, x_N$.

You are given a rooted tree with N vertices, and your task is to find a divisor labelling. The vertices are numbered from 1 to N, and 1 is the root.

Input

The first line contains an integer N ($1 \leq N \leq 60$).

The following $N - 1$ lines each contain two integers u and v ($1 \leq u, v \leq N$, $u \neq v$), meaning that an edge goes between vertices u and v. These edges will form a tree.

Output

Print one line with N integers, the numbers $x_1, x_2, \ldots x_N$. These numbers must satisfy $1 \leq x_i \leq 10^{18}$. It can be shown that under these constraints, an answer always exists.

Sample Input 1	Sample Output 1
5 1 2 1 3 3 4 3 5	1 2 3 21 33

This page is intentionally left blank.

Problem G
Groups of Strangers

Picture Image by macrovector at Freepik

A software company resulting from several recent mergers consists of a handful of offices in different cities. At each office there is a HR manager who knows everyone working at that office. Additionally, there are of course a lot of persons in the workforce that know each other, also across the offices. Knowing each other in this context is a symmetric relation, if employee a knows employee b, then b also knows a. In comparison, while everyone in the workforce (hopefully) knows *of* the CEO, that does not mean the CEO knows everyone. In fact, while the CEO *does* know all the HR managers, she does not know that many other employees. This has got to change!

One of the HR managers is instructed to plan a company outing to let people get to know each other better. Unfortunately, it seems impossible to a find a hotel that can accommodate the whole workforce. Maybe he can turn this into an advantage? He is thinking that by using up to three hotels in the vicinity of each other, perhaps it is possible to divide the employees in three groups, one per hotel, so that no two employees in a group already know each other? That would certainly encourage new acquaintances.

In this early stage in the planning, he doesn't care about the sizes of the groups, but only wants to see if such a division is at all possible. He asks you to help him and provides you with a complete list of pairs of employees who know each other. However, you are not told who the HR managers or CEO are, but you know that the CEO knows at most 15 employees (this includes the HR managers) and that the company is distributed over at most 8 offices.

Input

The input consists of one line with the integers N and M ($2 \leq N \leq 1000, 1 \leq M \leq 100000$), the number of employees and the number of pairs of employees that know each other. Next follow M lines each containing two integers a and b ($1 \leq a \neq b \leq N$), indicating that employee a knows employee b.

It is guaranteed that for the given test cases, there is at least one way to divide the employees into offices with HR managers and a CEO, like in the description. To summarize, the CEO is an employee who knows at most 15 other people. The rest of the employees can be partitioned into at most 8 offices, where each office has an employee (the HR manager) who knows all of the other employees in that office. In addition, the CEO knows all of the HR managers. Note that employees from different offices can know each other.

Output

If it is possible to divide the employees in at most three groups so that no pair that know each other are in the same group, output one row with the group identity $1, 2, 3$ for each employee in the same order they were numbered in the input. Use a single space between two subsequent employees' group identities. If multiple solutions exist, you may output any. If no such group partitioning exists, output "Impossible".

Sample Input 1

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

Sample Output 1

```
1 2 2 3
```

Sample Input 2

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

Sample Output 2

```
Impossible
```

Problem H
Heroes of Velmar

Welcome to the world of *Heroes of Velmar*, the critically acclaimed trading card game developed by *Sidney Games*! After the tremendous success of the physical card game, Sidney Games has decided to take it to the next level and transform it into an immersive video game experience.

As Sidney Games embarks on this ambitious video game project, they seek the expertise of talented developers like you to bring this digital version to life. The challenge lies in coding the algorithm that determines the winner in the virtual battles that unfold between players. The video game will need to retain the same core mechanics as the original card game, where players compete on three distinct locations over six turns, with abilities and power levels shaping the outcomes.

The full rules of the game are listed below. You are given the state of the locations after *Setup* and *Gameplay* have finished and the *End of the Game* has been reached. Sidney Games has tasked you with implementing the *Location Resolution* part of the game rules, including the application of *Special Abilities*, to determine the winner.

The game designers have provided you with images of the cards as well as a JSON file with their specifications.

Heroes of Velmar - Game Rules

Objective

The objective of the two-player card game *Heroes of Velmar* is to win more locations than the opponent over six turns. If there is a tie, the total power level across all locations acts as a tiebreaker.

Setup

1. Each player selects a deck of cards containing heroes with different abilities and power levels.
2. The players shuffle their decks and draw a starting hand of 7 cards each.
3. Designate three distinct locations for the battle. Each location will have its own separate battlefield.

Gameplay

1. Each player draws 1 card from the deck, to their hand, if possible. Maximum cards per hand is 7.
2. Players take turns playing cards from their hand on any of the three locations.
3. On each turn, players have energy equal to the turn number, for example, on Turn 3, they have 3 energy to spend.
4. Each card has a cost that must be paid using energy to play it on a location.
5. Players can play as many cards as they want as long as the following conditions are met:

(a) They have sufficient energy to spend on the card.

(b) The location has an empty spot. Each player has 4 available spots per location.

6. The power level of each card represents its strength in the game.
7. When the turn ends, the energy not used is lost.

End of the Game

1. The game ends after six turns, at which point locations are resolved, with each location having a winner or a tie.
2. The player who wins the most locations is the overall winner.
3. If the players win an equal number of locations, the total power level across all locations is used as a tiebreaker to determine the winner.
4. If the total power level across all locations is also tied, then the game results in a tie.

Location Resolution

1. Locations are resolved separately at the end of the game.
2. Compare the total power levels of all cards played by each player at the location after applying special abilities.
3. The player with the higher total power level wins the location.
4. If there is a tie, no one wins the location.

Special Abilities

1. Some cards have special abilities that can affect the game. These abilities trigger at the end of the game, before location resolution.
2. Abilities can buff the card's power level, interact with other cards, or manipulate the game state.
3. Two cards that portray the same character are considered to portray distinct characters for the sake of abilities.

The game *Heroes of Velmar* offers a mix of strategic card play, resource management, and tactical decision-making. Players must choose their heroes wisely, coordinate their plays, and employ their unique abilities to outwit their opponents and emerge victorious in the epic battle for control over Velmar!

Card design

Card Explanation:

- **Top Left Corner:** The energy cost required to play the card on the battlefield.
- **Top Right Corner:** The power level of the card, representing its strength.

- **Text on the Bottom:** The name of the card, identifying its character.
- **Text Underneath:** The card's ability, describing its unique effect during gameplay.

The cards are shown on the next page.

Input

Input consists of six lines representing the state of the three locations after six turns of play. This means that there will be at most 24 cards total listed in the input.

First the left location is described, then the center location is described, and finally the right location is described. Each location is described by two lines, the first of which represents player 1's cards and the second of which represents player 2's cards. Each line lists the number of cards in the line and then the names of the cards played by the player, separated by spaces. There will be at most four cards in each line.

Note that a player may leave a location empty. Each input is guaranteed to be a valid reachable final game state according to the game rules.

Output

Output `"Player 1"` if player 1 won, `"Player 2"` if player 2 won, or `"Tie"` if there was no victor.

Sample Input 1

```
3 Shadow Seraphina Ironwood
2 Voidclaw Voidclaw
1 Vexia
0
1 Ranger
0
```

Sample Output 1

```
Player 1
```

Sample Input 2

```
1 Guardian
1 Anvil
2 Seraphina Seraphina
1 Ranger
2 Ironwood Ranger
1 Guardian
```

Sample Output 2

```
Tie
```

Sample Input 3

```
1 Guardian
1 Anvil
2 Seraphina Seraphina
1 Ranger
1 Ironwood
1 Guardian
```

Sample Output 3

```
Player 2
```


4
6
SHADOW


3
5
GALE


2
4
RANGER


5
7
ANVIL


2
3
VEXIA


6
8
GUARDIAN


5
6
THUNDERHEART
Phalanx - If the location this card is played at has 4 friendly cards, including this one, then its power is doubled. Note that other buffs the card receives are not doubled.


1
2
FROSTWHISPER


1
3
VOIDCLAW


1
3
IRONWOOD


6
4
ZENITH
Centered focus - If this card is played at the center location, +5 power.


1
1
SERAPHINA
Celestial Healing - Seraphina grants +1 power to each friendly card at this location, other than itself. This includes other Seraphina cards.

Problem I
Intertwined

NCPC (Nordic Cargo Plane Control) are testing a new engine for their cargo planes. To this end they have bound a strong and sturdy infinitely thin rope to the centre of their testing platform, and to the engine. We will place a coordinate system onto this testing platform such that the rope is bound at the origin and lays along the positive x-axis to $(d, 0)$. On this testing platform there are also a number of infinitely thin pillars that can stop the rope, but ignore the engine. As the engine is started it starts rotating the rope counter-clockwise around the origin until it hits a pillar, at which point it is caught and starts rotating around that pillar counter-clockwise instead. The engine is then rotating at a smaller radius as some of the rope is caught between the origin and this pillar. This keeps going until the rope is too short to reach any other pillars.

Running these tests, buying all this infinitely thin rope and setting up these infinitely thin pillars, is expensive. Besides, the workers keep getting these nasty paper-like cuts from all these infinitely thin objects. It would be much more economical to just simulate the behaviour.

Input

The first line contains an integer n, the number of pillars, and an integer d, the length of the rope ($1 \leq n \leq 10^5$, $1 \leq d \leq 10^9$).

The following n lines each contain two integers x_i, y_i, the coordinates of the ith pillar ($-10^9 \leq x_i, y_i \leq 10^9$) for $i \in 1, 2, \ldots, n$. None of these pillars will lie on the rope.

Output

Print one line with an integer i, meaning that the rope will end up spinning around the ith pillar in the input. Note that this index is 1-indexed. If the rope doesn't collide with any pillars $i = -1$. It is guaranteed that changing the input d by at most $\pm 10^{-6}$ will not change i.

Sample Input 1	Sample Output 1
5 200 4 4 4 -4 3 1 -4 4 -4 -4	1

This page is intentionally left blank.

Problem J
Jamboree

Picture Image by brgfx at Freepik

A group of scouts are preparing to go to a large meeting with other scouts. Their leader Hildeborg, in spirit of the scout motto "be prepared", wants to distribute some useful items among the scouts that they most probably will need on their adventure. The items come in different sizes, so to make this as fair as possible, she wants to make sure that the total size of items carried by any scout is as small as possible. Furthermore, Hildeborg does not want to give more than two items to any scout as she is afraid that it otherwise will be too hard for them to remember to bring everything. Given the sizes of the items, what is the least maximum total size, computed as the sum of items, any scout will have to carry?

Input

The first line of input contains two positive integers N and M ($1 \leq N \leq 2M$, $1 \leq M \leq 100$). N is the number of useful items, and M is the number of scouts. The second line contains N positive integers a_i ($1 \leq a_i \leq 10^7$) giving the sizes of the items.

Output

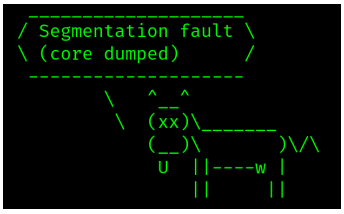
Print one integer, the smallest total size that any scout has to carry.

Sample Input 1	Sample Output 1
3 4 10 10 10	10

Sample Input 2	Sample Output 2
5 4 9 12 3 9 10	12

This page is intentionally left blank.

Problem K
Karl Coder



Karl is an aspiring C programmer, and is excited by the risks and rewards of low-level manual memory management. In the program he currently develops, he stores a string containing N non-zero bytes into a buffer named "`buf`". By mistake he accidentally made the buffer $2N$ bytes in size. The last N bytes of the buffer consists of only zero-bytes.

Now Karl needs to know the value N, the size of the string, in a separate part of the program. Traditionally you would recover the length of a string using the `strlen`-function, which reports the position of the first zero-byte in the provided buffer using a linear scan. However, Karl finds that this is much too slow, and that it defeats the advantage of using C in the first place. Can you help Karl efficiently recover N without crashing his program?

index	···	-1	0	1	2	3	4	5	6	7	8	···
buf	···	×	78	67	80	67	0	0	0	0	×	···

The contents of the buffer in sample interaction 3 are shown here.

Interaction

This is an interactive problem where interaction proceeds in rounds. In each round your program can attempt to read a byte from the buffer or report the answer (the value of N):

read: Write a line containing "`buf[`i`]`" if you want to try to read byte i of the buffer. If $0 \leq i < 2N$, then you will get the value of the byte stored in the buffer at index i. If $i < N$ then this will be an integer between 1 and 255, otherwise it will be 0. If $i \geq 2N$ or $i < 0$, you will get the response "`Segmentation fault (core dumped)`" and your program should exit.

answer: Write a line containing "`strlen(buf) =` M" if you want to report that you think that $N = M$. Afterwards your program should exit. Your submission will be accepted if you answered correctly, if you did not trigger a segmentation fault, and if you did not attempt to read entries from the buffer too many times.

A maximum of $2\lceil \log_2 N \rceil$ reads can be made. If your program attempts to read from the buffer after reaching the limit, it will get the response "`Too many reads`" and your program should then exit.

It is guaranteed that $2 \leq N \leq 10^{18}$.

Read	Sample Interaction 1	Write
	`buf[1]`	
`65`		
	`buf[2]`	
`0`		
	`strlen(buf) = 2`	

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

```
buf[1]
```

```
50
```

```
buf[5]
```

```
0
```

```
buf[7]
```

```
Segmentation fault (core dumped)
```

Read **Sample Interaction 3** **Write**

```
buf[0]
```

```
78
```

```
buf[1]
```

```
67
```

```
buf[2]
```

```
80
```

```
buf[3]
```

```
67
```

```
buf[4]
```

```
Too many reads
```