Problem C. Cake

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	1024 mebibytes

Once upon a time Babushka Bajtmiła (known for the exception pierogi from problem B) baked a cake. She cut it into $2n$ rectangular pieces (two rows containing n pieces each) and put colour icing on the top of each piece. She looked at her masterpiece and was taken aback: the colour scheme looked absolutely awful.

Bajtmiła decided that she needs to change the colour configuration to something better. Unfortunately, changing the locations of the pieces one by one is out of question: any attempt to take out a single piece out of the cake would inevitably make its sides chipped. Babushka Bajtmiła would never serve her guests a cake which appears raggedly cut!

Fortunately, Bajtmiła possesses a rectangular spatula which can fit exactly four pieces (in two rows containing two pieces each). She can therefore carefully take four such pieces out, rotate the spatula and insert them back from the opposite side. We can formally describe this operation as taking a 2 x 2 square and rotating it by 180 degrees.

Remembering your quick and efficient help in problem B, Babushka knew exactly what to do: she asked you to determine the minimum possible number of operations that would turn her initial frosting configuration into something nice. Of course, you did what any decent person would do in your place: you said you won't be able to help because the problem is not well-defined. Bajtmiła expected that: she went to the whiteboard and draw one specific *nice* configuration and asked you to find the minimal number of operations to change her initial frosting configuration into this specific one.

It might have also happened that the granny (tired from moving all the pierogi in problem B) was mistaken and it is actually impossible to obtain the desired configuration. In such case you also need to notify her as soon as possible!

Input

The first line of input contains the number of test cases z ($1 \leq z \leq 10\,000$). The descriptions of the test cases follow.

The first line of a test case contains an integer n ($2 \leq n \leq 500\,000$). The following two lines of input describe two rows of the initial frosting configuration. Each of them contains n integers from the range $[1, 10^9]$ separated by single spaces — identifiers of frosting colours on the subsequent pieces of the cake. Note that a colour may appear more than once.

The next two lines describe the final configuration in the same format.

The total sum of n over all the test cases does not exceed $2\,000\,000$.

Output

For each test case output a single integer — the minimum possible number of operations needed to achieve the final configuration. If it is not possible to achieve the desired outcome, print -1.

Example

standard input	standard output

2
4
1 2 3 2
4 3 1 3
3 2 1 1
2 3 4 3
2
1 2
3 4
3 4
1 2
	3
	-1

Note

In the first test case the initial configuration is

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

We can achieve the desired configuration in three steps:

1. rotating the leftmost square,

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

2. rotating the rightmost square,

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

3. rotating the middle square.

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

It is not possible to achieve the desired configuration in the second test case.

Problem D. Divided Mechanism

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

The Bytegate machine is Bajtazar's newest invention. It consists of two parts, which – to quote enthusiastic Bajtazar – "are simply impossible to split apart!"... or so he thought. His two-year-old son Bajtuś is just about to prove him wrong.

The initial state of mechanism can be described as an $n \times m$ array, filled with characters `A`, `B` and dots. Characters `A` correspond to the first part of mechanism, characters `B` – the second part of mechanism, and dots mean empty spaces:

```
+-+-+-+-+-+
|B|B|A|A|.|
+-+-+-+-+-+
|.|B|B|A|A|
+-+-+-+-+-+
|A|.|B|B|A|
+-+-+-+-+-+
|A|.|.|B|A|
+-+-+-+-+-+
|A|A|A|A|A|
+-+-+-+-+-+
```

Since the mechanism consists of two parts, the array contains at least one character `A` and at least one character `B`. Furthermore, both parts of the mechanism are connected, i.e. for any two A-cells there exists some path connecting these cells, passing only through other A-cells, with any two consecutive cells on this path sharing a common edge. The part B is connected in the same way.

Bajtuś plays with the machine by pulling part B in various directions. His play can be described as a sequence of q letters `N`, `S`, `E`, `W` – the directions of the consecutive moves (North, South, East and West, respectively). Each time, Bajtuś is pulling the component B in the chosen direction until any further pulling in that direction would cause the parts of the mechanism to overlap. If Bajtuś could continue that movement indefinitely, we say that the two parts have been split apart. It does not mean that Bajtuś stopped messing with the mechanism at that moment – nonetheless, once split, the mechanism remains split forever. Determine if Bajtuś will actually split the mechanism during his play.

Input

The first line of input contains the number of test cases z ($1 \leq z \leq 1\,000$). The descriptions of the test cases follow.

The first line of each test case contains three integers n, m, q ($1 \leq n, m \leq 10$, $1 \leq q \leq 100$). The next n lines describe the initial state of the mechanism. Each of these lines is a string of length m consisting of characters `A`, `B` and `.`. Both mechanism components are non-empty and connected.

The last line of each test case contains q characters belonging to the set $\{N, S, E, W\}$, describing Bajtuś's moves.

Output

For each test case print `TAK` if Bajtuś has split the mechanism. Otherwise, print a single word `NIE`.

Example

standard input	standard output
3	NIE
5 5 3	TAK
BBAA.	NIE
.BBAA	
A.BBA	
A..BA	
AAAAA	
WNW	
5 5 7	
BBAA.	
.BBAA	
A.BBA	
A..BA	
AAAAA	
WNWNSEN	
6 5 3	
.....	
.AAA.	
.A.A.	
.AB..	
.A.A.	
.AAA.	
SNE	

Note

The final states of mechanism in the first and the third test case are:

```
+-+-+-+-+-+-+-+
|B|B|.|.|.|.|.|
+-+-+-+-+-+-+-+
|.|B|B|.|A|A|.|
+-+-+-+-+-+-+-+
|.|.|B|B|.|A|A|
+-+-+-+-+-+-+-+
|.|.|A|B|.|.|A|
+-+-+-+-+-+-+-+
|.|.|A|.|.|.|A|
+-+-+-+-+-+-+-+
|.|.|A|A|A|A|A|
+-+-+-+-+-+-+-+

+-+-+-+
|A|A|A|
+-+-+-+
|A|B|A|
+-+-+-+
|A|.|.|
+-+-+-+
|A|.|A|
+-+-+-+
|A|A|A|
+-+-+-+
```

In the second test case Bajtuś splits the mechanism in the fourth step.

Problem F. Fence

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	1024 mebibytes

Bajtazar is widely believed to be the greatest scrooge in the whole borough. One can find many examples to support this claim, and the least important one of them is that his estate has not even got a fence. However, Bajtazar has recently found n old planks in his basement, so he decided to build at least a fragment of a hoarding.

He stacked the planks one over the other such that their consecutive lengths were $a_1, \ldots, a_n$. He took the first one, cut out a fragment of length b, and nailed it as the leftmost piece of the fence. Then, he cut out the next fragment of length b, and nailed it next to the previous one. He continued doing so until what was left in his hands was a piece of length c $(1 \le c \le b)$. *Well, this one might seem a little bit too short, but it would be a pity if such a good plank went to waste*, he thought... and added it to the fence as well. He then took the second plank from the stack, then the next one, and repeated the whole procedure for each one of them.

When the job was done, Bajtazar looked at the result... and concluded that using those shorter pieces might really not have been the best idea. *This doesn't look like a cohesive design at all* – he thought – and decided to paint the whole fence white to give it at least a pretence of consistency. *Still*, it occurred to him suddenly, *if I only paint every other plank white, and leave the remaining ones brown, I could use (roughly) twice as little paint and yet the fence will still look like a well thought-out, coherent construction!* And so he only painted every second plank, starting from the leftmost one (apparently, the rumours of Bajtazar's meanness must have been somewhat exaggerated. He could have started from the second leftmost plank after all.).

However, in the evening, a frightening thought struck him: maybe if he had chosen a different value of b, the overall amount of paint used could have been smaller? Well, there's not much that can be done anymore, but just the thought of such an unnecessary wastage keeps Bajtazar awake – he needs to know how much paint he would have used if he had chosen to build a fence of any other possible height. Help him to find the answer so he can finally fall asleep peacefully (or not, depending on the result of your computation).

Input

The first line of input contains the number of test cases z $(1 \le z \le 5)$.

The descriptions of the test cases follow.

The first line of every test case contains a single integer n $(1 \le n \le 1\,000\,000)$ – the number of planks. Then follow n integers a_i $(1 \le a_i \le 1\,000\,000, \sum_{i=1}^{n} a_i \le 1\,000\,000)$ – the lengths of consecutive planks.

Output

For each test case, output M lines, where M is the maximum of all values of a_i within this test case. The i-th line should contain a single integer f_i: the total length of planks which Bajtazar would have needed to paint white if he had chosen the fence height to be $b = i$.

Example

standard input	standard output

1
4
10 7 2 8
	14
	13
	15
	13
	15
	16
	21
	23
	24
	12

Note

For height $b = 4$, consecutive fence pickets would have heights:

4 4 2 4 3 2 4 4.

Bajtazar would have needed to paint the total length of $4+2+3+4$, so the answer in the 4-th line is 13.

For height $b = 5$, consecutive fence pickets would have heights:

5 5 5 2 2 5 3.

Bajtazar would have needed to paint the total length of $5+5+2+3$, so the answer in the 5-th line is 15.

Problem H. Hidden Password

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

After a successful internship, Bytholomew was hired as a senior cyber-security expert. To lead by example, he decided to finally heed his own advice and use two *different* passwords for his e-mail and for the Facepalm social media. Unfortunately, remembering two passwords proved too much for him. Moreover, he couldn't just plainly write the passwords somewhere, as it would be against another of his recommendations. But being a security expert, Bytholomew knew exactly what to do. He chose his favorite integer $d > 0$ and wrote both passwords encoded with Caesar cipher with key d.

Pleased with his work, he looked at his notes and the horrible truth dawned on him: after the encoding, the first (e-mail) password became literally the second (Facepalm) password, while the second one turned into the first one. ,,Holy moly!" – Bytholomew exclaimed, as there was nothing more to say.

Now you too can become a security expert – knowing the first of Bytholomew's passwords, guess the second one, if possible.

Input

The first line of input contains the number of test cases z ($1 \le z \le 20$). The descriptions of the test cases follow.

Each test case is one word – the password – consisting of lowercase English letters, in a separate line. The password has at least 1 character and at most 200 000 characters.

The total number of characters in all passwords does not exceed 1 000 000.

Output

For every given password, guess and output the second one in a separate line. If the second password cannot be determined (either because there is no solution or because there is more than one), output a single word `NIE` instead.

Example

standard input	standard output
1 cnffjbeq	password

Note

The Caesar cipher means substituting each letter with the one d places down the alphabet, treating alphabet as cyclic if necessary. E.g. for $d = 3$ the letter `a` is substituted by `d`, `b` by `e`,...., `w` changes into `z`, `x` to `a`, `y` to `b` and `z` to `c`.

Problem I. Interesting Numbers

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

The *bitwise exclusive or*, or simply XOR, is an operation denoted by $\oplus$ which works on two integers by XOR-ing their corresponding bits: if x_i, y_i, z_i denote the i-th binary digit of x, y and z, where $z = x \oplus y$, then $z_i = (x_i + y_i) \mod 2$.

You are given a positive integer k. A sequence of integers is *interesting* if XOR of any its two elements is less than or equal k.

Given a sequence $a_1, \ldots, a_n$, determine the largest possible length of its interesting subsequence. (A subsequence is a sequence that can be derived from the given sequence by deleting zero or more elements.)

Input

The first line of input contains the number of test cases z ($1 \leq z \leq 1\,000$). The descriptions of the test cases follow.

The first line of each case contains two integers n, k ($1 \leq n \leq 30\,000$, $1 \leq k < 2^{20}$) – the length of the sequence, and the upper bound on maximum XOR of its two elements.

The second line contains n nonnegative integers $a_1, \ldots a_n$ ($0 \leq a_i < 2^{20}$) – the given sequence described above.

The sum of n and k over all test cases do not exceed 200 000 and 3 200 000 respectively.

Output

For every test case output a single integer – the maximum possible length of an interesting subsequence of the given sequence.

Example

standard input	standard output

1
7 11
3 12 9 10 16 3 4
	4

Note

The elements 3, 9, 10 and 3 form an interesting subsequence, as XOR of every pair is not larger then 11. For example $9 \oplus 10 = 1001_2 \oplus 1010_2 = 11_2 = 3 \leq 11$. There is no subsequence consisting of five elements with the same property, e.g. the sequence $(3, 9, 10, 3, 4)$ is not interesting because of $4 \oplus 9 = 100_2 \oplus 1001_2 = 1101_2 = 13 > 11$.

Problem J. Jungle Trail

Input file:	*standard input*
Output file:	*standard output*
Time limit:	3 seconds
Memory limit:	1024 mebibytes

In the mobile game "Jungle Trail", you are given a rectangular $n \times m$ board divided into $n \cdot m$ squares. Each square is either empty, blocked (impassable) or contains a den of snakes, either poisonous or benign (not poisonous). If a square contains a den of snakes, then either all the snakes on a given field are poisonous, or all are benign.

The game allows you to tap any column or any row of the board. If you tap a column, all poisonous snakes in this column are turned to benign, and vice versa. Similarly, if you tap any row, all snakes in the row change their state. You can tap each row/column only once. If a den is in a tapped row as well as in a tapped column, its state returns to the original one.

After performing all those operations, you must find a trail through the jungle: a path which starts at the top left corner, in every move goes either one square down or one to the right, ends at the bottom right corner and never passes through a den of poisonous snakes or a blocked field.

Input

The first line of input contains the number of test cases z ($1 \le z \le 500$). The descriptions of the test cases follow.

The first line contains two integers n and m ($2 \le n, m \le 2\,000$).

Each of the following n lines contains m characters `.`, `#`, `O` (capital o) and `@` (at sign), meaning an empty square, blocked square, den of benign snakes and den of poisonous snakes, respectively. You may assume that the top left corner and the bottom right corner are not blocked.

Neither the sum of n values over all test cases nor the sum of all m values exceed $15\,000$.

Output

For every test case output the solution in the following format:

The first line should contain `TAK` if a jungle trail is possible or `NIE` if it isn't.

If the answer is `TAK`, in the next three lines output:

- A sequence of n characters T or N, the i-th character being T if the i-th row should be tapped, N if not; - A sequence of m characters T or N, determining in the same way whether the columns should be tapped; - A sequence of $n+m-2$ characters P or D denoting the trail: P means a move right, D means a move down.

Example

standard input	standard output
1 4 5 ..#.. @@O@@ ##@#O ..@.@	TAK NTNN NNTNT DPPDDPP

Note

After tapping the rows and columns described on the output, the board is in the following state:

```
..#..
```

```
OOOO@
##O#@
..O.O
```

Now the given path goes only through `.` and `O` squares.

Problem K. Kitten and Roomba

Input file:	`standard input`
Output file:	`standard output`
Time limit:	15 seconds
Memory limit:	1024 mebibytes

A little kitten of Bituś, Kapitan, does love sleeping! Sadly, the average quality of Kapitan's sleep has decreased significantly after its owner decided to buy Roomba – a robot vacuum cleaner. It seems the kitten is scared of Roomba as... well, it is fairly scared.

Bituś's house consists of n rooms connected by $n-1$ two-way corridors in such a way that it is possible to reach any room from any other one. Bituś noticed that whenever Roomba enters a room with Kapitan inside, the kitten awakes instantly and runs away to one of the neighbouring rooms, where it returns back to dreaming about playing with mice. Frightened Kapitan escapes the room blindly, so if there exists more than one room directly connected to the current one, then *every neighbouring room is equally likely to chosen by Kapitan* (in particular, it can escape to the room from which Roomba has just came from).

During one particularly long night shift at work, Bituś opened the Roomba app and observed that during today's cleaning it visited rooms $a_1, \ldots, a_m$ (in this order). A room can appear more than once in this sequence, but every two neighbouring rooms must be directly connected to each other. Bituś also remembers that initially the kitten had been sleeping in the room c. Moreover, it must hold that $a_1 \neq c$ because observant Kapitan would never ever sleep in one room with Roomba!

Now Bituś wonders what is the *expected value* of the number of times Roomba has woken Kapitan during the cleaning session. Please help Bituś to find the answer so he can finally return back to work.

Input

The first line of input contains the number of test cases z ($1 \leq z \leq 6\,000$). The descriptions of the test cases follow.

The first line of a test case contains two integers n, c ($2 \leq n \leq 1\,000\,000$, $1 \leq c \leq n$), the number of rooms in Bituś's house and the identifier of the room where Kapitan sleeps initially.

The following $n-1$ describe corridors. Each of them contains two integers u_i, v_i ($1 \leq u_i, v_i \leq n$, $u_i \neq v_i$) signifying that the rooms u_i and v_i are connected. You can assume that you can reach every room from any other.

The next line contains the number of rooms m ($1 \leq m \leq 5\,000\,000$) visited by Roomba during the cleaning session.

The last line of the test case contains a sequence of m integers a_i ($1 \leq a_i \leq n$) – the rooms visited (in this order) by Roomba. Every two subsequent rooms are connected with a corridor; moreover, assume that $a_1 \neq c$.

The sum of values of $n + m$ over all test cases does not exceed $12\,000\,000$.

Output

For every test case print one real number e – the expected number of times Roomba entered a room with Kapitan inside. Your answer will be considered correct if its absolute or relative error does not exceed 10^{-5}. Namely, if your answer is a, and the correct value is b, then your answer will be accepted if $\frac{|a-b|}{\max(1,b)} \leq 10^{-5}$.

Example

standard input	standard output

1
4 2
1 2
2 3
4 2
4
1 2 3 2
	1.666666666666667

Problem L. Lemurs

Input file:	*standard input*
Output file:	*standard output*
Time limit:	4 seconds
Memory limit:	1024 mebibytes

Bajtazar has been sent on a mission to the jungle. He is going to study the behaviour of newly discovered species – taxicab lemurs.

The jungle has a rectangular shape with dimensions n by m. Taxicab lemurs are already known to be group animals. Furthermore, each group of lemurs inhabits only one cell of the jungle, i.e. a single cell with coordinates (x, y), where $1 \le x \le n$ and $1 \le y \le m$. But the most fascinating aspect of their behaviour lies in their foraging area: if some group of lemurs inhabits a cell with coordinates (x, y), then their foraging area is bounded by a ball with radius k in the taxicab metric. In other words, the foraging area of this group of lemurs is the set of cells with coordinates (x', y') satisfying $|x - x'| + |y - y'| \le k$ and $1 \le x' \le n$, $1 \le y' \le m$. The constant k is universal for all taxicab lemurs.

The last taxicab lemur's researcher has disappeared under unknown circumstances and the only thing he left behind was an n by m map with some cells marked as taxicab lemurs' foraging area (the circumstances of this event are still unclear; the latest theory suggests that the newly discovered species of lemurs may not be herbivorous after all...). Slightly disturbed by that story, Bajtazar is currently wondering if this map is reliable. Therefore he would like to check if there exists such a set of groups of taxicab lemurs, that their foraging area would match precisely the cells marked on the map.

Help Bajtazar and find the answer to his bugging question. He will certainly be grateful till his very last days!

Input

The first line of input contains the number of test cases z ($1 \le z \le 4\,000$). The descriptions of the test cases follow.

The first line of each test case consists of numbers n, m, k ($1 \le n, m, k \le 1\,000$) with meaning as described in the task statement.

The next n lines describe the map. If, according to the map, the taxicab lemurs forage on a cell with coordinates (j, i), then in the i-th line on j-th position will be a character `x`, otherwise this character will be a dot.

The sum $n + m + k$ over all the test cases does not exceed $100\,000$.

Output

For each test case print a single line. If the answer to Bajtazar's question is yes, print `TAK`, otherwise print `NIE`.

Example

standard input	standard output
2 3 3 1 .xx xxx xx. 3 4 1 ..xx x.xx x..x	TAK NIE

Note

In the first test case, the marked foraging area is generated by a set of 3 groups of lemurs inhabiting cells with coordinates $(1, 3)$, $(2, 2)$ and $(3, 1)$.

In the second test case, there does not exists such a set of groups of lemurs that would generate the foraging area described by the map.

Problem N. Express As The Sum

Input file:	*standard input*
Output file:	*standard output*
Time limit:	2 seconds
Memory limit:	512 mebibytes

Given an integer N, express it as the sum of at least two consecutive positive integers. For example:

$10 = 1 + 2 + 3 + 4$

$24 = 7 + 8 + 9$

If there are multiple solutions, output the one with the smallest possible number of summands.

Input

The first line of input contains the number of test cases T. The descriptions of the test cases follow:

Each test case consists of one line containing an integer N ($1 \leq N \leq 10^9$).

Output

For each test case, output a single line containing the equation in the format:

```
N = a + (a+1) + ...+ b
```

as in the example. If there is no solution, output a single word "IMPOSSIBLE" instead.

Examples

standard input	standard output

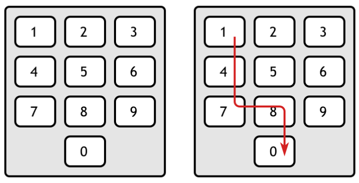
Problem O. Keyboard Troubles

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	512 mebibytes

Good morning! This is your 5am wake-up call! A partly cloudy day is expected with light rain coming afternoon...

You have just woken up. You desperately need coffee... and... more coffee... and some cereal. And your clothes. And coffee.

To prepare warm cereal, you put some milk into a microwave, trying to heat it for k seconds. You must enter k on the microwave keyboard:



As you still haven't had your coffee, your hand (along with eyes and brain) keeps falling down. You are only able to enter a number if your hand would only move downwards and/or to the right. You cannot go back left, nor move your hand up, though you can press the same key again. And again... and again...

For example, you can enter the number 180 or 49, but not 98 or 132. Enter a number that is as close to k as possible. If there are two solutions, enter any one of them. You are too sleepy to actually care. And you need coffee.

Input

The first line of input contains the number of test cases T. The descriptions of the test cases follow:

Each test case consists of one line containing an integer k ($1 \leq k \leq 200$).

Output

For each test case, output a number that is closest to k which can be entered on the keyboard.

Examples

standard input	standard output
3 180 83 132	180 80 133

Problem P. The Exam

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 mebibytes

Professor Byteoni is preparing *Bit & Byte Theory* exam. He has already prepared n questions. Each of these questions has been ranked with an expected difficulty coefficient by the professor. This coefficient is a natural number ranging from 1 to n. Each of the questions holds a different coefficient.

Now the professor is considering the exam questions sequence. Professor wishes to determine whether his students are able to judge the question difficulty by themselves. For this purpose he plans to line up his questions in such a way, that coefficients of subsequent questions differ at least by k. Help the professor to find such a sequence.

Input

The first and only input line contains two integers n and k ($2 \leq n \leq 1\,000\,000$, $1 \leq k \leq n$): the number of questions prepared by professor and the lower limit of the difficulty difference of subsequent exam questions.

Output

Your program should output one line containing sought question difficulty coefficients sequence, in other words a sequence of n pairwise distinct natural numbers ranging from 1 to n, where each two subsequent numbers differ at least by k. If there are numerous correct answers, your program should write any one of these. In case the sought sequence does not exist, your program should write only one word: "`NIE`" (Polish for *no*).

Examples

`standard input`	`standard output`

Problem Q. Robin Hood

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	512 mebibytes

Robin Hood takes the rich to distribute to the poor. Together with his gang they robbed a convoy carrying gold to the counts' castle and n caskets fell prey to robbers. After transporting their loot to the cave it turned out that i-th (for $i = 1, 2, \ldots, n$) casket contains exactly i money-bags full of gold.

In case a poor man comes to Robin Hood asking for a few gold ducats, Robin Hood utilises the following procedure. Firstly he chooses a non-empty casket that contains the smallest number of money-bags containing gold. In case the casket contains exactly only one money-bag, Robin Hood hands it to the man in need, and sees him go away happily. Otherwise, if the casket contains an odd number of money-bags, Robin Hood puts one of the money-bags in his pocket, and starts the whole process again. However, in case there is an even number of money-bags, Robin Hood takes exactly half of them out and puts them in an empty casket (luckily the empty caskets are plentiful in the cave) and begins the whole procedure anew. Therefore if a penniless man comes to Robin Hood, and in case he still will be in a possession of at least one non-empty casket, as a result of (possibly multiple) employment of Robin Hood's procedure, the poor man is sure to get the money-bag full of gold. The poor would come to the Robin Hood's cave until all the caskets are empty.

Fellow robbers from Robin Hood's gang wonder if their leader does not ruin the good name of thugs with his behaviour. They want to know how many looted money-bags remain in Robin Hood's pocket when all the caskets are empty.

Input

The first and only line of the input contains one integer n ($1 \leq n \leq 10^9$), which indicates the number of caskets robbed by Robin Hood's gang.

Output

The first and only line of output should contain an integer representing the number of money-bags with gold, which will remain in Robin Hood's pocket after emptying all the caskets.

Examples

standard input	standard output

Problem R. Unused Digits

Input file:	*standard input*
Output file:	*standard output*
Time limit:	1 second
Memory limit:	256 mebibytes

Given a string, consisting of digits from 0 to 9. Find out all digits, which are not used in this string.

Input

First line of the input file contains one integer T — number of the test cases ($1 \le T \le 150$). Each of next T lines contains one test case — non-empty string, consisting of digits from 0 to 9. Length of the given string does not exceed 30.

Output

For each test case print in accending order all digits, which did not used in this string. If all ten digits are used, print "`full`" instead.

Examples

standard input	standard output
2 2468 0123456789	013579 full