Problem A. Assembly

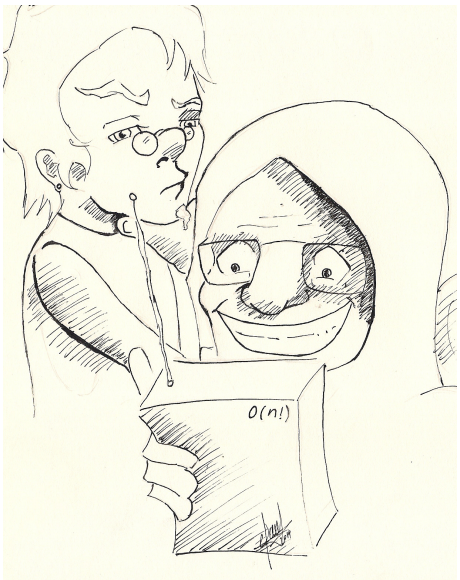
Source file name:	Assembly.c, Assembly.cpp, Assembly.java, Assembly.py
Input:	`Standard`
Output:	`Standard`

This year some Colombian university have decided to remodel the access to the university, and for giving them a modern touch, the Superior Council of this university chose to put some control access devices for avoiding strangers and thiefs to make harm to students and facilities.

In this university there are a lot of left wing political groups, and the main feature of this kind of groups is to be against almost any decision taken by the Superior Council, and the decision about control access devices will not be the exception, obviously. For deciding wich actions will be taken against the Superior Council decisions, these political students group gather all the other students in an event called student assembly, which you can modelate in the following way:

- There is an speech sequence of size N and N speaker students which are the ones who persuade the other students to take position against or in favor of an initiative from Superior Council. Each speech in the sequence corresponds to only one speaker student.
- For each speech i there is an acceptance level $S(0 \leq S \leq 100)$ after it. If $S = 0$, then, all the students are against the superior council decision, but if $S = 100$, then, all the students will accept the superior council decision.

The university Headmaster has found out about students intentions. For that reason, he decided to sneak in the students assembly disguised as a revolutionary student, and not only that, but also he carried with him a sensor designed by the enigneering school which allows him to know in real time each speech acceptance level.



You as a clever student of the computer science program find out the mean intentions of the headmaster, so, after some minutes of hacking the new headmaster's toy, you realize that the sensor works in the following way:

- The device will show to the headmaster a message saying "Caution", if the longest increasing subsequence of speeches is equally larger than the longest decreasing subsequence of speeches.

- The device will show to the headmaster a message saying "Don't worry", if the longest increasing subsequence of speeches is strictly less or strictly greater than the longest decreasing subsequence.
- An increasing subsequence is defined as a subset(not necesarily continuous) $S^* = [a_1, a_2, \ldots, a_k]$ from the original sequence of speeches $S = [S_0, S_1, \ldots, S_N]$, such that if $i > j$, then, $a_i > a_j$. Here a_i is acceptance level for an speech within the subsequence, and S_i is the acceptance level in the original sequence.
- A decreasing subsequence is defined as a subset(not necesarily continuous) $S^* = [b_1, b_2, \ldots, b_k]$ from the original sequence of speeches $S = [S_0, S_1, \ldots, S_N]$,such that if $i > j$, then $b_i < b_j$. Here b_i is the acceptance level in the subsequence.

For example, supose you have the next sequence of speeches $[10, 25, 20, 22, 90, 21]$, one increasing subsequence could be $[10, 90]$ and one decreasing subsequence could be $[25, 20]$ but this both are not necesearily the longest ones. In this case the longest increasing subsequence is $[10, 20, 22, 90]$ and the longest decreasing subsequence is $[25, 22, 21]$, look that in this case the output message from the sensor will be "Don't worry".

After knowing this, you realize that the sensor used by the headmaster have an algorithm for finding this subsequences that runs in $O(N!)$ time, and while you are thinking why some profesors of the engineering school that designed de sensor are so lazy, you also wonder if you can design a better algorithm to know before the headmaster what is the outcome of the sensor.

Given that outcome you will decide if you must or must not intervene in the assembly, either for avoiding a strike or avoiding people to forget the university issues.

Hint: given a number in the original sequence you can know easily whether the numbers before the given number are less or greater than it, and store that info in someway. take care about the number of operations, and take into account that sometimes there are data already calculated that maybe you do not need to calculate again.

Input

The input will have several test cases, in the first line of each test case there will be an integer $N(1 \leq N \leq 20000)$ corresponding to the amount of speeches. The second line will have N integer numbers $S_1,\ S_2,\ \ldots,\ S_i,\ S_N$, where S_i $(0 \leq s_i \leq 100)$ represents the acceptance level for the speech i.

Output

There are only two options of output

- If the longest increasing subsequence is equal in length as the longest decreasing subsequence you must print "Caution. I will not intervene." representing the outcome of the sensor and what you will probably do.
- If the longest increasing subsequence is NOT equal in length as the longest decreasing subsequence you must print "Don't wory. I must intervene."

Example

Input	Output
6 10 25 20 22 90 91 5 10 10 10 10 10 5 1 20 50 49 48	Don't worry. I must intervene. Caution. I will not intervene. Caution. I will not intervene.

Problem B. A Puzzle from Red Matemática

Source file name:	Puzzle.c, Puzzle.cpp, Puzzle.java, Puzzle.py
Input:	`Standard`
Output:	`Standard`

On January 29, 2014, Red Matemática proposed the following mathematical challenge on their twitter account (@redmatematicant): "Natalia wrote a number, repeating the digit six 2014 times, and then multiplied that number by itself. She calls Daniel and asks him: what is the sum of the digits of this square?"

Using this interesting puzzle as our starting point, the problem you are asked to solve now is: given a digit d ($1 \leq d \leq 9$), form a number n by repeating d exactly k times, where $1 \leq k \leq 100000$, and then calculate n^2. Finally, answer the question: what is the sum of the digits of this square?

Input

The input may contain several test cases. Each test case is presented on a single line, and contains two positive integers d and k. The input ends with a test case in which both d and k are zero, and this case must not be processed.

Output

For each test case in the input, your program must print the positive integer that represents the sum of the digits of n^2. Each valid test case from the input must generate a single line of output.

Example

Input	Output
1 1	1
6 2014	18126
1 1000	8992
9 10000	90000
0 0	

Problem C. Tobby and Goldbach's Conjecture

Source file name:	Goldbach.c, Goldbach.cpp, Goldbach.java, Goldbach.py
Input:	`Standard`
Output:	`Standard`

Tobby is a smart dog who loves coding. Tobby is fascinated by Goldbach's conjecture, a cojeture very easy to describe but which has never been proven, despite efforts to do so by the best mathematicians in history. The conjecture states the following:

"Every even integer greater than 2 can be written as the sum of to prime numbers."

To learn how to code Tobby wants to solve a related problem: given an integer N, in how many different ways can that number be written as the sum of two prime numbers?

To solve the problem Tobby prepared a list of all the prime numbers less than or equal to 400:

2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101, 103, 107, 109, 113, 127, 131, 137, 139, 149, 151, 157, 163, 167, 173, 179, 181, 191, 193, 197, 199, 211, 223, 227, 229, 233, 239, 241, 251, 257, 263, 269, 271, 277, 281, 283, 293, 307, 311, 313, 317, 331, 337, 347, 349, 353, 359, 367, 373, 379, 383, 389, 397

Input

The first line contains an integer A ($1 \leq A \leq 200$), the there are A lines, each one with an even integer N ($4 \leq N \leq 400$)

Output

For each integer N in the input, you should write a line with the number of different ways that N can be written as the sum of two prime numbers.

Example

Input	Output
3 4 10 22	1 2 3

Problem D. Numeric Center

Source file name:	Center.c, Center.cpp, Center.java, Center.py
Input:	`Standard`
Output:	`Standard`

A numeric center is a number that separates in a consecutive and positive integer number list (starting at one) in two groups of consecutive and positive integer numbers, in which their sum is the same. The first numeric center is number 6, which takes the list $\{1,\ 2,\ 3,\ 4,\ 5,\ 6,\ 7,\ 8\}$ and produces two lists of consecutive and positive integer numbers in which their sum (in this case 15) is the same. Those lists are: $\{1,\ 2,\ 3,\ 4,\ 5\}$ and $\{7,\ 8\}$. The second numeric center is 35, that takes the list $\{1,\ 2,\ 3,\ 4, \ldots,\ 49\}$ and produces the following two lists: $\{1,\ 2,\ 3,\ 4, \ldots,\ 34\}$ and $\{36,\ 37,\ 38,\ 39, \ldots,\ 49\}$, the sum of each list is equal to 595.

The task consists in writing a program that calculates the total of numeric centers between 1 and n.

Input

The input consists of several test cases. There is only one line for each test case. This line contains a positive integer number n ($1 \leq n \leq 10^{14}$). The last test case is a value of n equal to zero, this test case should not be processed.

Output

For each test case you have to print in one line, the number of numeric centers between 1 and n.

Example

Input	Output
1	0
7	0
8	1
48	1
49	2
50	2
0	

Problem E. Tobby and the Primoshkas' Tree

Source file name:	Primoshkas.c, Primoshkas.cpp, Primoshkas.java, Primoshkas.py
Input:	`Standard`
Output:	`Standard`

Tobby just found a very special tree. This kind of tree produces a magical creature called Primoshka. Primoshkas are little and quite funny creatures that entertain people in several ways. The operation of this tree is a little weird, First of all, if someone wants to collect Primoshkas he/she must select a fork in the tree and tell it one of the following phrases:

- "Please create new primoshkas", in this case for each fork below the selected fork (the selected fork is affected too), a new primoshka arises. In case that one primoshka already exists in a determined fork, she falls and dissapears.
- "Please collect primoshkas", in this case all primoshkas below to the current fork (and also in the selected fork) are collected, however, due to unknown reasons a bunch of new primoshkas appear after that, but only in the forks where primoshkas previously existed (Leaving the tree as if nothing had changed).

Initially the tree is empty.

The tree just have one root, in others words, there are exactly one fork which have no forks above it. Furthermore the tree don't have any kind of cycles.

Input

The input consists of several test cases. The test case begins with two numbers N and Q followed by $N-1$ lines with a pair of numbers u_i, v_i which indicate that the fork number u_i and the fork number v_i are connected (v_i is below u_i).

After that, Q lines with a pair of integers op_i and f_i which indicates the type of the operation (0 for the first phrase and 1 for the second phrase) and the number of the selected fork.

Constraints:

Small cases: $3 \leq N \leq 100$ and $1 \leq Q \leq 100$

All forks are labeled between 0 and $N-1$

Output

For each one of the second type operations you must print the number of collected primoshkas in that operation

Example

Input	Output
5 4 0 1 1 2 1 3 0 4 0 1 1 1 0 0 1 0	3 2

Problem F. Square Formations

Source file name:	Formations.c, Formations.cpp, Formations.java, Formations.py
Input:	`Standard`
Output:	`Standard`

Raúl and Jaime, as you may recall, are two young and bright brothers who love games. Jaime has received as a present a set of little plastic soldiers, and he happily shares it with Raúl every time they get together to play.

Their greatest source of amusement comes from arranging their groups of soldiers in different ways. Raúl, for instance, likes to vary his formation patterns and arrange his platoons in circular or elliptic shapes, while Jaime undoubtedly prefers square formations for his soldiers–in which the number of rows is equal to the number of columns.

One day, Raúl organizes a surprise attack using two platoons of soldiers, attacking two opposing flanks. This baffles Jaime, who after a long pause to think about it, discovers that when he divides his square formation into two, it becomes impossible to reassemble the halves in new square formations, and this happens regardless of the number of soldiers in his original formation.

Raúl pauses to think about this too, and eventually presents Jaime with an alternative: he can prepare a first group of N soldiers (N being a positive, even integer), and maintain two auxiliary groups of K soldiers. If N and K are chosen carefully, Raúl explains, then it is possible to create square formations with $N + K$ and $(N/2) + K$ soldiers.

Jaime likes the elegance of this idea, and doesn't take long to discover an example that illustrates Raúl's suggestion: if he keeps a main group of 14 soldiers, and two auxiliary groups with 2 soldiers each, then he can use one of the auxiliary groups to produce a main formation of 16 soldiers, and if he needs to split his main platoon into two, then he can produce two minor square formations of 9 soldiers. So, $N = 14$, $K = 2$ is a valid choice for Jaime.

According to this, we'll say that a pair of integers (N, K) is valid if it holds for Raúl's proposition; that is, if it is possible to produce square formations with groups of $N + K$ and $(N/2) + K$ soldiers. The question in their minds now is: given an arbitrary value of K, what would be the lowest possible N that generates a valid pair?

Input

Input starts with a positive integer T, that denotes the number of test cases.

Each test case is described by a single integer K in its own line. It can be safely assumed that all test cases have a solution in which N is a number not greater than 10^6.

$T \leq 1000$; $1 \leq K \leq 1000$

Output

For each test case, print the case number, followed by the value of N. This number must be the lowest integer possible, such that (N, K) forms a valid pair according to Raúl's proposition.

Example

Input	Output
3 1 2 36	Case 1: 48 Case 2: 14 Case 3: 1728

Problem G. Tobby and the River

Source file name:	River.c, River.cpp, River.java, River.py
Input:	`Standard`
Output:	`Standard`

Tobby is in one side of a river with 5 other dogs. In that side there are two small boats in which Tobby and the other five dogs can cross to the other side. Tobby knows that in order to cross to the other side the weight in both boats must be as balanced as possible, because if it is not the boats might sink.

Tobby wants to know if there is a way to divide the 6 dogs (he and the other 5 dogs) in both boats in such a way that both boats have the same weight. Both boats don't need to have the same amount of dogs, but all the 6 dogs must be in one of the two boats.

Input

In the first line there is a number A ($1 \leq A \leq 100$), after that there are A lines with 6 positive integer numbers less than or equal to 100.

Output

For each one of the input lines you must write a line with the spanish sentence "Tobby puede cruzar" if there is a way of dividing the 6 dogs in both boats with the same weight or the spanish sentence "Tobby no puede cruzar" otherwise.

Example

Input	Output
4 1 3 3 2 1 2 6 3 2 4 5 1 3 3 3 3 3 3 1 1 1 1 1 5	Tobby puede cruzar Tobby no puede cruzar Tobby puede cruzar Tobby puede cruzar

Problem H. Humbertov's Flag

Source file name:	Humbertov.c, Humbertov.cpp, Humbertov.java, Humbertov.py
Input:	`Standard`
Output:	`Standard`

Humbertov Moralov wants to create a new flag for his empire, probably because he is thinking to invade Tobyland again. He doesn't want a common rectangular flag and he has ordered you to find new designs. You have to show Moralov a good amount of different designs quiclky or Moralov is going to ask for your head.

The new designs should be polygons, not only convex polygons, but you certainly don't want self intersecting polygons because you can't translate them to real flag designs. So you decide to write a program to generate non self-intersecting polygons, so you decided to write a program that generates these polygons for you. Given a set of points, your program must output a non self-intersecting polygon that includes every one of the given points.

Input

The input consists to several test cases. Each test case begins with a integer N indicating the number of points in the list. Then come N lines each with two integers indicating the x and y coordinates of each point of the list.

- $3 \leq N \leq 100$
- $-100000 \leq x_i, y_i \leq 100000$

Output

For each test case, print N followed by N lines each with one of the points read on the input, but you can change the order in any way you like such that the resulting polygon is not self intersecting.

Example

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

Problem I. Tiki-Taka

Source file name:	Tikitaka.c, Tikitaka.cpp, Tikitaka.java, Tikitaka.py
Input:	`Standard`
Output:	`Standard`

Football, like any other sport, is continually evolving, and there have been teams which have developed truly marvelous playing styles throughout the years. Among the most cited examples, you could point to the national football teams of Brazil, Netherlands and Argentina during the 70s, as well as to many European clubs that have won glorious titles in recent decades, such as Dutch club Ajax, Milan of Italy, Barcelona of Spain and Germany's Bayern München.

One of the most popular styles used in modern football is the so-called *tiki-taka*, which generally refers to a particular style in which short passes and movements are critical, and the overarching goal is to maintain possession of the ball, wearing out the opposing team and disrupting their defensive lines.

The football manager (FM) of the national team for the country of Nlogonia–a beautiful place where football is gaining popularity–is planning a tiki-taka training session with a group of his players. He knows that, within the framework of this simple style, there is an ample range of play combinations, and he wants to explore those possibilities. For this purpose, he starts by giving out instructions to his players, indicating the possible *passing lines*. A passing line is defined by two players; one who passes the ball, and one who receives the pass. The FM expressly forbids passing the ball from a player A to another player B unless he has defined a passing line from A to B.

The FM wants to prepare a special play with N passes, and he chooses two players, which will be denoted X and Y, to be the starting and ending points of the play–that is, X must make the first pass, and Y has to receive the last pass. Now, the FM wants to know the number of different ways in which such play can be executed, performing N passes, starting with X and ending in Y. Nlogonia's FM is so busy these days with the team that he asks your help to create a program that calculates this value for him.

Input

Input starts with a positive integer T ($T \leq 20$), denoting the number of test cases.

Every test case starts with a blank line. The next line contains two integers: P, L, representing the number of players to be part of the training session, and the number of passing lines defined by the FM, respectively.

The next L lines describe the passing lines. Each of these lines contain two integers: A, B, indicating that player A may pass the ball to player B ($A \neq B$). Take notice that this does not necessarily imply that B is allowed to pass the ball to A. No passing line will appear more than once in the input.

The next line contains an integer Q, the number of queries made by the FM. The next Q lines describe the queries. Each of these lines contain three integers: X, Y, N, in that order.

$2 \leq P \leq 11$; $1 \leq Q \leq 30$; $1 \leq N \leq 10^{15}$; $1 \leq A, B, X, Y \leq P$

Output

For each test case, your program must print a line with the message **Case i:**, replacing **i** with the case number. Then you must print Q lines, with the answers to the queries, in the same order as they were presented in the input.

The answer to each query must be the total number of different ways in which the players can pass the ball N times, starting with X and ending in Y. Since this number can be very large, print this answer modulo 1000000007.

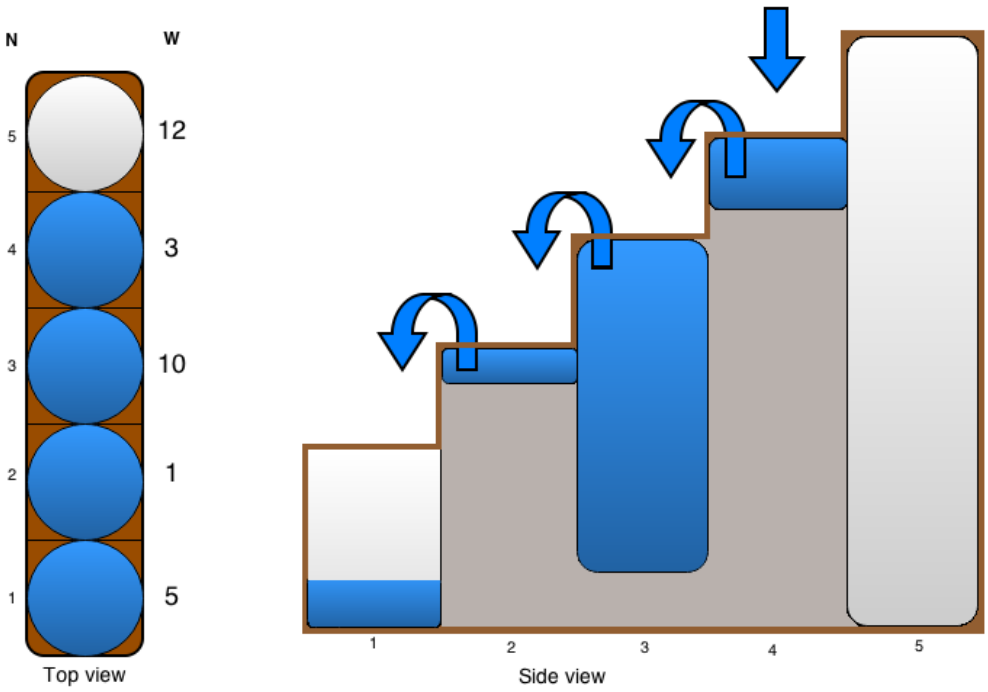
Example

Input	Output
2 4 6 1 2 2 1 2 3 2 4 3 2 3 4 4 1 4 2 1 4 4 2 4 10 4 1 1 3 4 1 2 2 1 2 3 3 2 1 1 3 30	Case 1: 1 2 16 0 Case 2: 16384

Problem J. Tobby and Tanks

Source file name:	Tanks.c, Tanks.cpp, Tanks.java, Tanks.py
Input:	`Standard`
Output:	`Standard`

Tobby is a curious dog and he is always thinking about new and revolutionary ideas. Now he imagined N tanks placed in a row (staggered - See the figure). Every tank has a capacity of W liters of water. The cute and curious dog is asked himself, if you have K liters of water, which is the i-th tank (where i is the maximum possible), such that if we drop the water there, the water will reach the first tank. As you know Tobby is not a complicated dog, so he does not want completely fill the first tank, he will be happy if the first tank have at least 1 liter of water.



You are given n tanks with capacity w_i, and q queries, each query contains one integer k, the amount of water.

Input

For every test case, the first line contains two integers n ($1 \leq n \leq 10^5$) and q ($1 \leq q \leq 10^4$), the number of tanks and the number of queries respectively, the next line contains n integers w_i ($1 \leq w \leq 10^4$). The next q lines contains a single integer k ($1 \leq k \leq 10^9$), the amount of water. (Read until EOF).

Output

For every query you must print a single integer, the maximum i ($1 \leq i \leq n$)

Example

Input	Output
5 2 5 1 10 3 12 16 1	4 1

Problem K. Tobby and the Divisors - I

Source file name:	Divisors01.c, Divisors01.cpp, Divisors01.java, Divisors01.py
Input:	`Standard`
Output:	`Standard`

Tobby is a smart dog who loves coding. Tobby has always been told that the most important part of coding is doing class diagrams and that algorithmic complexity is irrelevant. Santiago, Tobby's teacher, wants to teach Tobby the importance of algorithmic complexity with a simple problem:

"Given an integer number N, what is the positive integer number less than or equal to N that has the most divisors? If there are several integers less than or equal to N with the maximum amount of divisors, what is the smallest integer among them?"

Santiago gave Tobby an example: when N is equal to three, both two and three have two divisors and therefore the answer is two, being the smallest one.

Santiago also told Tobby the following:

"The number of divisors of an integer M is the amount of integers between 1 and M that divide M exactly. That is, the number of integers X between 1 and M such that $M\%X = 0$."

Input

The first line contains an integer A (such that $1 \leq A \leq 50$). Then there are A lines, each one with an integer N (such that $1 \leq N \leq 2500$).

Output

For each number N in the input you must print the positive integer less than or equal to N that has the highest number of divisors. If there are several numbers less than or equal to N with the maximum number of divisors then you must print the smallest integer among them.

Example

Input	Output
3 3 20 2000	2 12 1680

Problem L. Lineland

Source file name:	Lineland.c, Lineland.cpp, Lineland.java, Lineland.py
Input:	`Standard`
Output:	`Standard`

Lineland has always been caracterized as a prolific civilization but in this moment, thanks to what seems to be an attack of an individual, The electrical network from lineland has been damaged, they don't know where the responsible came from, maybe from flataland, from cubicland or even tobyland. Well, the fact is that now the city managers from lineland only can use one electrical plant from from the N possible electrical plants installed in lineland. So they have to choose an electrical plant which is going to be able to give energy to maximum amount of cities. can you help the managers from lineland with this problem?.

Lineland can be modelate as a set of points on an horizontal axis, each point will represent an electrical plant ubicated in one city. Besides, one electrical plant ubicated in a point x_i with a capacity c_i, will be able to supply a city ubicated in x_j if, and only if, $x_i < x_j < x_i + c_i$. Your task is to choose the electrical plant which can suply the maximum number of cities.

Input

The input will have several test cases, in the first line of each test case there will be an integer $N(1 \leq N \leq 10^6)$, In the next N lines there will be two integers x and c $(0 \leq x \leq 10^9, 0 \leq c \leq 10^9)$; x_i represents the position of the $i-th$ city and the $i-th$ plant, while c_i represents the capacity of the $i-th$ plant $(1 \leq i \leq N)$. You might assume this two tips:

- No two cities i and j are ubicated in the same location.
- For each two cities ubicated in x_i, $x_j (i! = j)$ with capacity c_i and c_j that can supply another cities within a range of $(x_i, x_i + c_i)$ and $(x_j, x_j + c_j)$ respectively, this two conditions hold: $x_i + c_i ! = x_j + c_j$ and $x_i + c_i ! = x_j$.

Output

You have to print two integers in one line, that is, the position of the electric plant that will supply the maximum amount of cities, followed by an space and the respective amount of cities that will be supplied by that electric plant.

Example

Input	Output
7 1 7 5 4 6 5 7 15 10 3 12 5 16 3 5 1 1 3 2 6 3 10 4 15 22 3 1000 1000000000 1003 10000 50000 1000000000	1 3 1 0 1000 2

Problem M. Tobby and the Divisors - II

Source file name:	Divisors02.c, Divisors02.cpp, Divisors02.java, Divisors02.py
Input:	`Standard`
Output:	`Standard`

Tobby is a smart dog who loves coding. Tobby has always been told that the most important part of coding is doing class diagrams and that algorithmic complexity is irrelevant. Santiago, Tobby's teacher, wants to teach Tobby the importance of algorithmic complexity with a simple problem:

"Given an integer number N, what is the positive integer number less than or equal to N that has the most divisors? If there are several integers less than or equal to N with the maximum amount of divisors, what is the smallest integer among them?"

Santiago gave Tobby an example: when N is equal to three, both two and three have two divisors and therefore the answer is two, being the smallest one.

Santiago also told Tobby the following:

"The number of divisors of an integer M is the amount of integers between 1 and M that divide M exactly. That is, the number of integers X between 1 and M such that $M\%X = 0$."

Santiago then asked Tobby for the answer to the problem with numbers N less than or equal to 2500.

After Tobby solved that version of the problem, Santiago asked Tobby for the answer to the problem with numbers N less than or equal to 1000000. Tobby tried to solve the problem using the same code used for the easier version of the problem and got a "Time Limit" verdict. Tobby then asked Santiago for clues about how to solve this new version of the problem and Santiago told him:

"If we start counting from 1 to N, how many numbers less than or equal to N divide each number? 1 divides all the N numbers, 2 divide about $N/2$ numbers, 3 divides about $N/3$ numbers, and so on."

Tobby then realized that if we wanted to know for each number between 1 and N how many numbers divide each one of them, we could start a count array in 0 for all numbers and then we could add 1 to the count of all the N numbers which 1 divides, then add 1 to the count of all the approximately $N/2$ numbers which 2 divides, and so on.

Tobby then asked himself: how many times would I end up adding to the count array? He realized that the answer to his question is the solution to the equation: sum for all i between 1 and N of $1/N$.

Input

The first line contains an integer A (such that $1 \leq A \leq 50$). Then there are A lines, each one with an integer N (such that $1 \leq N \leq 1000000$).

Output

For each number N in the input you must print the positive integer less than or equal to N that has the highest number of divisors. If there are several numbers less than or equal to N with the maximum number of divisors then you must print the smallest integer among them.

Example

Input	Output
3 3 20 2000	2 12 1680

Problem N. Tobby and the Divisors - III

Source file name:	Divisors03.c, Divisors03.cpp, Divisors03.java, Divisors03.py
Input:	`Standard`
Output:	`Standard`

Tobby is a smart dog who loves coding. Tobby has always been told that the most important part of coding is doing class diagrams and that algorithmic complexity is irrelevant. Santiago, Tobby's teacher, wants to teach Tobby the importance of algorithmic complexity with a simple problem:

"Given an integer number N, what is the positive integer number less than or equal to N that has the most divisors? If there are several integers less than or equal to N with the maximum amount of divisors, what is the smallest integer among them?"

Santiago gave Tobby an example: when N is equal to three, both two and three have two divisors and therefore the answer is two, being the smallest one.

Santiago also told Tobby the following:

"The number of divisors of an integer M is the amount of integers between 1 and M that divide M exactly. That is, the number of integers X between 1 and M such that $M\%X = 0$."

Santiago then asked Tobby for the answer to the problem with numbers N less than or equal to 2500.

After Tobby solved that version of the problem, Santiago asked Tobby for the answer to the problem with numbers N less than or equal to 1000000. Tobby tried to solve the problem using the same code used for the easier version of the problem and got a "Time Limit" verdict. Tobby then asked Santiago for clues about how to solve this new version of the problem and Santiago told him:

"If we start counting from 1 to N, how many numbers less than or equal to N divide each number? 1 divides all the N numbers, 2 divide about $N/2$ numbers, 3 divides about $N/3$ numbers, and so on."

Tobby then realized that if we wanted to know for each number between 1 and N how many numbers divide each one of them, we could start a count array in 0 for all numbers and then we could add 1 to the count of all the N numbers which 1 divides, then add 1 to the count of all the approximately $N/2$ numbers which 2 divides, and so on.

Tobby then asked himself: how many times would I end up adding to the count array? He realized that the answer to his question is the solution to the equation: sum for all i between 1 and N of $1/N$.

Santiago then asked Tobby for the answer to the problem with numbers N less than or equal to 1000000.

After thinking a lot, Tobby was finally able to solve that second version of the problem, but then Santiago asked Tobby for the answer to the problem with numbers N less than or equal to 10^{16}. Tobby tried to solve the problem using the same code used for the second version of the problem and got a "Time Limit" verdict. Tobby then asked Santiago for clues about how to solve this new version of the problem and Santiago told him:

"The answer to this problem is always a number belonging to the set of the highly composite numbers."

Tobby searched on the internet about the highly composite numbers and found the following:

"A highly composite number is a positive integer with more divisors than any positive integers smaller than himself."

If $N = p_1^{a_2} \text{ x } p_2^{a_3} \text{ x } \cdots \text{ x } p^{a_p}$ is the prime factorization of a highly composite number (ordered from smallest to biggest prime) then the following is true:

1. The primes $p_1, p_2, \ldots, p$ are consecutive primes starting in 2.
2. The exponents never increase: $a_2 \geq a_3 \geq \cdots \geq a_p$".

Tobby also found in the library of his univesity a guide by Humbertov Moralov that said the following about the relationship between the prime factorization of a number and the number of divisors of the same number:

"If the prime factorization of a number N is: $p_1^{a_1}$ x $p_2^{a_2}$ x $\cdots$ x $p_x^{a_x}$, where $p_1, p_2 \cdots p_n$ are distinct primes and $a_1, a_2 \cdots a_n$ are exponents greater than or equal to 1, then the number of divisors of N is: $(a_1 + 1)$ x $(a_2 + 1)$ x $\cdots$ x $(a_d + 1)$"

Tobby showed this information to Santiago, but he told him that he thought that there were many numbers up to 10^{16} which fullfill the conditions 1 and 2 given before. Santiago then gave Tobby a last clue:

"Last year in a coding competition I was not able to solve a problem which I could have solved generating all numbers that fullfilled those two conditions. After the competitions I thought about the problem and I finally realized that the numbers to generate where less than 100000 ... and the maximum value of N in that problem was 10^{18}, that is, 100 times the maximum value of N in this problem."

Input

The first line contains an integer A (such that $1 \leq A \leq 50$). Then there are A lines, each one with an integer N (such that $1 \leq N \leq 10^{16}$).

Output

For each number N in the input you must print the positive integer less than or equal to N that has the highest number of divisors. If there are several numbers less than or equal to N with the maximum number of divisors then you must print the smallest integer among them.

Example

Input	Output
3 3 20 2000	2 12 1680