Problem A. Average

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	512 megabytes

We often compute the average as the first step in processing statistical data. Yes, the average is a good tendency measure of data, but it is not always the best. In some cases, the average may hinder the understanding of the data.

For example, consider the national income of a country. As the term income inequality suggests, a small number of people earn a good portion of the gross national income in many countries. In such cases, the average income computes much higher than the income of the vast majority. It is not appropriate to regard the average as the income of typical people.

Let us observe the above-mentioned phenomenon in some concrete data. Incomes of n people, $a_1, \ldots, a_n$, are given. You are asked to write a program that reports the number of people whose incomes are less than or equal to the average $\frac{a_1+\ldots+a_n}{n}$.

Input

The first line of the input contains an integer T, which denotes the number of datasets.

Then follows T datasets, each in the following format.

n
$a_1\ a_2 \ldots a_n$

A dataset consists of two lines. In the first line, the number of people n is given. n is an integer satisfying $2 \le n \le 10\,000$. In the second line, incomes of n people are given. a_i $(1 \le i \le n)$ is the income of the i-th person. This value is an integer greater than or equal to 1 and less than or equal to 100 000.

The end of the input is indicated by a line containing a zero. The sum of n's of all the datasets does not exceed 50 000.

Output

For each dataset, output the number of people whose incomes are less than or equal to the average.

Example

stdin	stdout

5
7
15 15 15 15 15 15 15
4
10 20 30 60
10
1 1 1 1 1 1 1 1 1 100
7
90 90 90 90 90 90 10
7
2 7 1 8 2 8 4
	7
	3
	9
	1
	4

Problem B. Boxed Lunch

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	512 megabytes

Taro has been hooked on making lunch boxes recently. Taro has obtained a new lunch box recipe book today, and wants to try as many of the recipes listed in the book as possible.

Enough of the ingredients for all the recipes are at hand, but they all are in vacuum packs of two. If only one of them is used leaving the other, the leftover will be rotten easily, but making two of the same recipe is not interesting. Thus he decided to make a set of lunch boxes, each with different recipe, that wouldn't leave any unused ingredients.

Note that the book may include recipes for different lunch boxes made of the same set of ingredients.

How many lunch box recipes can Taro try today at most following his dogma?

Input

The first line contains n, which is the number of recipes listed in the book, and m, which is the number of ingredients. Both n and m are positive integers and satisfy $1 \le n, m \le 500$ and $1 \le n \cdot m \le 500$.

The following n lines contain the information for each recipe with the string of length m consisting of 0 or 1. $b_{i,j}$ implies whether the i-th recipe needs the j-th ingredient. 1 means the ingredient is needed for the recipe and 0 means not. Each line contains at least one 1.

Output

Output the maximum number of recipes Taro can try.

Example

stdin	stdout

4 3
110
101
011
110
	3

5 1
1
1
1
1
1
	4

4 5
10000
01000
00100
00010
	0

6 6
111111
011000
100000
000010
100001
100100
	6

Problem C. Cut Tree

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	512 megabytes

Given a tree T with n nodes, how many subtrees T' of T have at most K edges connected to $(T - T')$?

Input

The first line contains two integers n and K ($2 \le n \le 50$, $1 \le K \le n$).

The following $n - 1$ lines each contains two integers a and b denoting that there's an edge between a and b ($1 \le a, b \le n$).

Every node is indicated by a distinct number from 1 to n.

Output

A single integer which denotes the number of possible subtrees.

Note

Subtrees must be connected, but can be possibly empty.

Example

stdin	stdout

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

Explanation

There are 2^3 possible sub-trees: $\varnothing, \{1\}, \{2\}, \{3\}, \{1, 2\}, \{1, 3\}, \{2, 3\}, \{1, 2, 3\}$.

But: the sub-trees $\{2\}$ and $\{1, 3\}$ are not valid. $\{2\}$ isn't valid because it has 2 edges connecting to it's complement $\{1, 3\}$ whereas $K = 1$ in the sample test-case. $\{1, 3\}$ isn't valid because, well, it's not a sub-tree. The nodes aren't connected.

Problem D. DP Killer

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	512 megabytes

A typical dynamic programming problem would be like this: you are given a grid with h rows and w columns. In each cell there exists a number. Your task is to walk from $(1, 1)$ to (h, w) and obtain the largest possible sum.

However the round writer didn't want a dynamic programming problem, so he (she?) somehow modified the problem so that you might try something else.

Now you have a grid with h rows and w columns. $h + w$ **is even**. We denote the cell at the i-th row from the top and the j-th column from the left by (i, j). In any cell (i, j), an integer between 1 and 9 is written if $i + j$ is even, and either '`+`' or '`*`' is written if $i + j$ is odd.

You can get a mathematical expression by moving right or down $h + w - 2$ times from $(1, 1)$ to (h, w) and concatenating all the characters written in the cells you passed in order. Your task is to maximize the calculated value of the resulting mathematical expression by choosing an arbitrary path from $(1, 1)$ to (h, w). If the maximum value **is** 10^{15} **or less**, print the value. Otherwise, print -1.

Input

The first line consists of two integers h and w ($1 \leq h, w \leq 50$). It is guaranteed that $h + w$ is even.

The following h lines represent the characters on the grid. $a_{i,j}$ represents the character written in the cell (i, j). In any cell (i, j), an integer between 1 and 9 is written if $i + j$ is even, and either '`+`' or '`*`' is written if $i + j$ is odd.

Example

stdin	stdout

`3 3`
`1+2`
`+9*`
`1*5`
	`46`

`1 31`
`9*9*9*9*9*9*9*9*9*9*9*9*9*9*9*9`
	`-1`

`5 5`
`2+2+1`
`+1+1+`
`1+2+2`
`+1+1+`
`1+1+2`
	`10`

`9 7`
`8+9*4*8`
`*5*2+3+`
`1*3*2*2`
`*5*1+9+`
`1+2*2*2`
`*3*6*2*`
`7*7+6*5`
`*5+7*2+`
`3+3*6+8`
	`86408`

Problem E. Endless BFS

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	512 megabytes

Mr. Endo wanted to write the code that performs breadth-first search (BFS), which is a search algorithm to explore all vertices on an undirected graph. An example of pseudo code of BFS is as follows:

Algorithm 1 Typical BFS

```
$current \leftarrow \{\text{StartVertex}\}$
$visited \leftarrow current$
while $visited \neq$ the set of all the vertices do
    $found \leftarrow \varnothing$
    for all $v \in current$ do
        for all $u$ adjacent to $v$ do
            $found \leftarrow found \cup \{u\}$
        end for
    end for
    $current \leftarrow found \setminus visited$
    $visited \leftarrow visited \cup found$
end while
```

However, Mr. Endo apparently forgot to manage visited vertices in his code. More precisely, he wrote the following code:

Algorithm 2 Mr. Endo's BFS

```
$current \leftarrow \{\text{StartVertex}\}$
while $current \neq$ the set of all the vertices do
    $found \leftarrow \varnothing$
    for all $v \in current$ do
        for all $u$ adjacent to $v$ do
            $found \leftarrow found \cup \{u\}$
        end for
    end for
    $current \leftarrow found$
end while
```

You may notice that for some graphs, Mr. Endo's program will not stop because it keeps running infinitely. Notice that it does not necessarily mean the program cannot explore all the vertices within finite steps. See example 2 below for more details. Your task here is to make a program that determines whether Mr. Endo's program will stop within finite steps for a given graph in order to point out the bug to him. Also, calculate the minimum number of loop iterations required for the program to stop if it is finite.

Input

The first line consists of two integers n ($2 \leq n \leq 10^5$) and m ($1 \leq m \leq 10^5$), where n is the number of vertices and m is the number of edges in a given undirected graph, respectively.

The i-th line of the following m lines consists of two integers u_i, v_i ($1 \leq u_i, v_i \leq n$), which means the vertices u_i and v_i are adjacent in the given graph. The vertex 1 is the start vertex, i.e. StartVertex in the pseudo codes. You can assume that the given graph also meets the following conditions.

- The graph has no self-loop, i.e., $u_i \neq v_i$ for all $1 \leq i \leq m$.

- The graph has no multi-edge, i.e., $\{u_i, v_i\} \neq \{u_j, v_j\}$ for all $1 \leq i < j \leq m$.
- The graph is connected, i.e., there is at least one path from u to v (and vice versa) for all vertices $1 \leq u, v \leq n$.

Output

If Mr. Endo's wrong BFS code cannot stop within finite steps for the given input graph, print -1 in a line. Otherwise, print the minimum number of loop iterations required to stop.

Example

stdin	stdout

3 3
1 2
1 3
2 3
	2

4 3
1 2
2 3
3 4
	-1

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

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

Explanation

Explanation to example 2: transition of *current* is $\{1\} \rightarrow \{2\} \rightarrow \{1,3\} \rightarrow \{2,4\} \rightarrow \{1,3\} \rightarrow \{2,4\} \rightarrow \cdots$ Although Mr. Endo's program will achieve to visit all the vertices (in 3 steps), current will never become the same set as all the vertices.

Problem F. Find Palindromic Subsets

Input file:	`stdin`
Output file:	`stdout`
Time limit:	3 seconds
Memory limit:	512 megabytes

Consider a lowercase English alphabetic letter character denoted by c. A shift operation on some c turns it into the next letter in the alphabet. For example, and $shift(a) = b$, $shift(e) = f$, $shift(z) = a$.

Given a zero-indexed string, s, of n lowercase letters, perform q queries on s where each query takes one of the following two forms:

- $1\ i\ j\ t$: All letters in the inclusive range from i to j are shifted t times.
- $2\ i\ j$: Consider all indices in the inclusive range from i to j. Find the number of non-empty subsets of characters, $c_1, c_2, \ldots, c_k$ where $i \leq$ index of $c_1 <$ index of $c_2 < \cdots <$ index of $c_k \leq j$, such that characters $c_1, c_2, c_3, \ldots, c_k$ can be rearranged to form a palindrome. Then print this number modulo $10^9 + 7$ on a new line. **Two palindromic subsets are considered to be different if their component characters came from different indices in the original string.**

Input

The first line contains two space-separated integers describing the respective values of n and q.

The second line contains a string of n lowercase English alphabetic letters (i.e., a through z) denoting s. Each of the q subsequent lines describes a query in one of the two formats defined above.

Output

For each query of type 2 (i.e., $2\ i\ j$), print the number of non-empty subsets of characters satisfying the conditions given above, modulo $10^9 + 7$, on a new line.

Constraints

- $1 \leq n \leq 10^5$.
- $1 \leq q \leq 10^5$.
- $0 \leq i \leq j < n$ for each query.
- $0 \leq t \leq 10^9$ for each query of type 1.

Example

stdin	stdout
3 5 aba 2 0 2 2 0 0 2 1 2 1 0 1 1 2 0 2	5 1 2 3

Explanation

We perform the following $q = 5$ queries:

1. 2 0 2: $s = aba$ and we want to find the palindromic subsets of substring aba. There are five such subsets that form palindromic strings (a, b, a, aa, and aba), so we print the result of 5 mod $(10^9 + 7) = 5$ on a new line.

2. 2 0 0: $s = aba$ and we want to find the palindromic subsets of substring a. Because this substring only has one letter, we only have one subset forming a palindromic string (a). We then print the result of 1 mod $(10^9 + 7) = 1$ on a new line.

3. 2 1 2: $s = aba$ and we want to find the palindromic subsets of substring ba. There are two such subsets that form palindromic strings (b and a), so we print the result of 2 mod $(10^9 + 7) = 2$ on a new line.

4. 1 0 1 1: $s = aba$ and we need to perform $t = 1$ shift operations on each character from index $i = 0$ to index $j = 1$. After performing these shifts, $s = bca$.

5. 2 0 2: $s = bca$ and we want to find the palindromic subsets of substring bca. There are three valid subsets that form palindromic strings (b, c, and a), so we print the result of 3 mod $(10^9 + 7) = 3$ on a new line.

Problem G. Gnomes

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	512 megabytes

A family of n gnomes likes to line up for a group picture. Each gnome can be uniquely identified by a number $1, \ldots, n$ written on their hat.

Suppose there are 5 gnomes. The gnomes could line up like so: $1, 3, 4, 2, 5$.

Now, an evil magician will remove some of the gnomes from the lineup and wipe your memory of the order of the gnomes. The result is a subsequence, perhaps like so: $1, 4, 2$.

He then tells you that if you ordered all permutations of $1, \ldots, n$ in lexicographical order, the original sequence of gnomes is the first such permutation which contains the remaining subsequence. Your task is to find the original sequence of gnomes.

Input

The input will begin with a line with two integers n and then m ($1 \le m \le n \le 10^5$), where n is the number of gnomes originally, and m is the number of gnomes remaining after the evil magician pulls his trick. Each of the next m lines will contain a single integer g ($1 \le g \le n$). These are the remaining gnomes, in order. The values of g are guaranteed to be unique.

Output

Output n lines, each containing a single integer, representing the first permutation of gnomes that could contain the remaining gnomes in order.

Example

stdin	stdout

5 3
1
4
2
	1
	3
	4
	2
	5

7 4
6
4
2
1
	3
	5
	6
	4
	2
	1
	7

Problem H. Here Comes A Math Problem

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	512 megabytes

Arthur defines a function, $f(k)$, to be the number of (p, q) pairs such that:

- $1 < p \leq q \leq k$
- p and q are coprime.
- $p \cdot q = k$

Given an integer, n, help Arthur find and print the result of:

$$\sum_{k=1}^{n} f(k)$$

Input

The first line contains a single integer denoting $n (1 \leq n \leq 10^9)$.

Output

Print the result of $\sum_{k=1}^{n} f(k)$ on a new line.

Example

stdin	stdout
`12`	`3`

Explanation

The value of $f(k)$ for $1 \leq k \leq 12$ is:

- For $k = 6$, there is only 1 valid pair, $(2, 3)$, so $f(6) = 1$.
- For $k = 10$, there is only 1 valid pair, $(2, 5)$, so $f(10) = 1$.
- For $k = 12$, there is only 1 valid pair, $(3, 4)$, so $f(12) = 1$.

For all other $1 \leq k \leq 12$, the function returns 0. Thus, our final sum is the result of $1 + 1 + 1 = 3$.

Problem I. Intersections

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	512 megabytes

Intersections of Crossing Path City are aligned to a grid. There are N east-west streets which are numbered from 1 to N, from north to south. There are also N north-south streets which are numbered from 1 to N, from west to east. Every pair of east-west and north-south streets has an intersection; therefore there are N^2 intersections which are numbered from 1 to N^2.

Surprisingly, all of the residents in the city are Ninja. To prevent outsiders from knowing their locations, the numbering of intersections is shuffled. You know the connections between the intersections and try to deduce their positions from the information. If there are more than one possible set of positions, you can output any of them.

Input

The first line consists of an integer N ($2 \leq N \leq 100$). The following $2N^2 - 2N$ lines represent connections between intersections.

The $(i+1)$-th line consists of two integers a_i and b_i ($1 \leq a_i, b_i \leq N^2, a_i \neq b_i$), which represent that the a_i-th and b_i-th intersections are adjacent. More precisely, let's denote by (r, c) the intersection of the r-th east-west street and the c-th north-south street. If the intersection number of (r, c) is a_i for some r and c, then the intersection number of either $(r-1, c)$, $(r+1, c)$, $(r, c-1)$ or $(r, c+1)$ must be b_i. All inputs of adjacencies are different, i.e., $(a_i, b_i) \neq (a_j, b_j)$ and $(a_i, b_i) \neq (b_j, a_j)$ for all $1 \leq i < j \leq 2N^2 - 2N$. This means that you are given information of all adjacencies on the grid.

The input is guaranteed to describe a valid map.

Output

Print a possible set of positions of the intersections. More precisely, the output consists of N lines each of which has space-separated N integers. The c-th integer of the r-th line should be the intersection number of (r, c).

If there are more than one possible set of positions, you can output any of them.

Example

stdin	stdout

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

Two more examples on the next page.

stdin	stdout
3	1 2 3
1 2	4 6 5
4 7	7 8 9
8 6	
2 3	
8 9	
5 3	
4 6	
5 6	
7 8	
1 4	
2 6	
5 9	
4	8 2 5 7
12 1	3 13 14 10
3 8	11 15 6 1
10 7	16 4 9 12
13 14	
8 2	
9 12	
6 14	
11 3	
3 13	
1 10	
11 15	
4 15	
4 9	
14 10	
5 7	
2 5	
6 1	
14 5	
16 11	
15 6	
15 13	
9 6	
16 4	
13 2	

Problem J. Jim and the Skyscrapers

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	512 megabytes

Jim has invented a new flying object called HZ42. HZ42 is like a broom and can only fly horizontally, independent of the environment. One day, Jim started his flight from Dubai's highest skyscraper, traveled some distance and landed on another skyscraper of same height! So much fun! But unfortunately, new skyscrapers have been built recently.

Let us describe the problem in one dimensional space. We have in total N skyscrapers aligned from left to right. The i^{th} skyscraper has a height of h_i. A flying route can be described as (i, j) with $i \neq j$, which means, Jim starts his HZ42 at the top of the skyscraper i and lands on the skyscraper j. Since HZ42 can only fly horizontally, Jim will remain at the height h_i only. Thus the path (i, j) can be valid, only if each of the skyscrapers $i, i+1, \ldots, j-1, j$ is not strictly greater than h_i and if the height of the skyscraper he starts from and arrives on have the same height. Formally, (i, j) is valid iff $\nexists k \in [i, j] : h_k > h_i$ and $h_i = h_j$.

Help Jim in counting the number of valid paths represented by ordered pairs (i, j).

Input

The first line contains N, the number of skyscrapers. The next line contains N space separated integers representing the heights of the skyscrapers.

$1 \leq N \leq 3 \cdot 10^5$ and no skyscraper will have height greater than 10^6 and less than 1.

Output

Print an integer that denotes the number of valid routes.

Example

stdin	stdout
6 3 2 1 2 3 3	8
3 1 1000 1	0

Explanation

First testcase: (1, 5), (1, 6) (5, 6) and (2, 4) and the routes in the opposite directions are the only valid routes.

Second testcase: (1, 3) and (3, 1) could have been valid, if there wasn't a big skyscraper with height 1000 between them.

Problem K. KBlack Playing Cards

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	512 megabytes

KBlack has recently brought back a deck of cards from Gambia. He believes that playing cards during programming contests can protect his eyesight, especially when he has 10 problems unsolved with 4 hours left.

Ultmaster: "Are we playing cards, eh?"

KBlack: "Well ok then."

Ultmaster: "Watch this now. You have a deck of N cards, where the i^{th} card has a claim written on it in the form of '**at least** a_i cards beneath this one contain a false claim.' You have to shuffle them so that exactly K cards contain a false claim."

KBlack: "You destroy me in this game every time, where did you get these cards, mate?!"

Ultmaster: "Ah, my old man organizes card tournaments, so each day he gives me free cards, ten bucks per deck."

Your task is to help KBlack solve Ultmaster's task. Output the order in which KBlack must place the cards. It is also possible that this is a bad joke on Ultmaster's part, and that there is no possible order to place the cards in. In that case, output -1.

*Ultmaster's comment: The claim is '**at least** a_i cards beneath this one contain a false claim'. When a claim is false, it is lying about something, so the truth is: **less than** a_i cards beneath this one contain a false claim.'*

Input

The first line contains integers N and K ($1 \le N \le 5 \cdot 10^5$, $0 \le K \le N$).

The i^{th} of the following N lines contains an integer a_i ($0 \le a_i \le 5 \cdot 10^5$).

Output

If the required order does not exist, output -1.

Otherwise, in a single line, output N integers separated by spaces, the numbers on the cards in the order from the top to the bottom of the deck. If there are multiple solutions, output any.

Example

stdin	stdout

4 2
1
2
2
3
	2 3 1 2

5 3
2
1
3
0
3
	3 3 0 1 2

6 4
0
2
5
2
0
1
	-1

Explanation

Here is the clarification of the second test case.

For simplicity's sake, we'll label the cards as true/false, depending on the claims written on them.

Beneath the fifth card there are 0 false cards, so it is false.

Beneath the fourth card there is 1 false card, so it is true.

Beneath the third card there is 1 false card, so it is true.

Beneath the second card there is 1 false card, so it is false.

Beneath the first card there are 2 false cards, so it is false.

Therefore, we have a total of 3 false cards.

Problem L. Lena Sort

Input file:	`stdin`
Output file:	`stdout`
Time limit:	2 seconds
Memory limit:	512 megabytes

Lena developed a sorting algorithm described by the following pseudocode:

```
lena_sort(array nums) {
    if (nums.size <= 1) {
        return nums;
    }
    pivot = nums[0];
    array less;
    array more;
    for (i = 1; i < nums.size; ++i) {
        // Comparison
        if (nums[i] < pivot) {
            less.append(nums[i]);
        }
        else {
            more.append(nums[i]);
        }
    }
    sorted_less = lena_sort(less);
    sorted_more = lena_sort(more);
    ans = sorted_less + pivot + sorted_more;

    return ans;
}
```

We consider a comparison to be any time some `nums[i]` is compared with `pivot`.

You must solve q queries where each query i consists of some len_i and c_i. For each query, construct an array of len_i distinct elements in the inclusive range between 1 and 10^9 that will be sorted by `lena_sort` in exactly c_i comparisons, then print each respective element of the unsorted array as a single line of len_i space-separated integers; if no such array exists, print -1 instead.

Input

The first line contains a single integer denoting q (the number of queries). Each line i of the q subsequent lines contains two space-separated integers describing the respective values of len_i (the length of the array) and c_i (the number of comparisons) for query i.

- $1 \le q \le 10^5$
- $1 \le len_i \le 10^5$
- $0 \le c_i \le 10^9$
- $\sum_i len_i \le 10^6$

Output

Print the answer to each query on a new line. For each query , print space-separated integers describing each respective element in an unsorted array that Lena's algorithm will sort in exactly c_i comparisons; if no such array exists, print -1 instead.

Example

stdin	stdout

2
5 6
5 100
	4 2 1 3 5
	-1

3
1 0
4 6
3 2
	1
	4 3 2 1
	2 1 3

Explanation

For example 1:

We perform the following q queries:

One array with $len = 5$ elements is $[4, 2, 1, 3, 5]$. The sequence of sorting operations looks like this:

Run `lena_sort` on $[4, 2, 1, 3, 5]$.

Compare $pivot = 4$ with $2, 1, 3$, and 5 for a total of 4 comparisons. We're then left with $less = [2, 1, 3]$ and $more = [5]$; we only need to continue sorting less, as more is sorted with respect to itself because it only contains one element.

Run `lena_sort` on $less = [2, 1, 3]$.

Compare $pivot = 2$ with 1 and 3 for a total of 2 comparisons. We're then left with $less = [1]$ and $more = [3]$, so we stop sorting. We sorted $[4, 2, 1, 3, 5]$ in $4 + 2 = 6$ comparisons and $c = 6$, so we print `4 2 1 3 5` on a new line.

It's not possible to construct an array with $len = 5$ elements that `lena_sort` will sort in exactly 100 comparisons, so we print -1 on a new line.

Problem M. Memory Penalty

Input file:	`stdin`
Output file:	`stdout`
Time limit:	1 second
Memory limit:	512 megabytes

ACM (Abnormal Chip Manufacturer) has produced a new memory chip. This memory chip has a large number of blocks, so large that you can assume it's infinite. What is a little bizarre about this chip, however, is that the blocks are numbered $1, 2, 3, \dots$ and the penalty to transfer data from block $\#i$ is i nanoseconds.

As is well-known, in computer science, a data block, for example, an array, is usually stored in continuous blocks, i.e., from l to r. And the cost to transfer all these data from memory to the processor is $(l + (l + 1) + \cdots + r)$ nanoseconds.

Now you are given the total penalty to transfer a data block, please find out l and r.

If you don't have time to read the statement above, in other words, you are given an integer n, and you are asked to find some positive integers l and r $(1 \le l \le r)$ such that $\sum_{i=l}^{r} i = n$.

Input

The first line contains an integer T $(1 \le T \le 1000)$ indicating the number of test cases.

Then follows T lines, each line contains an integer n $(2 \le n \le 10^9)$.

Output

For each n, output l and r, separated with space, in one line. If **there are multiple solutions, you should output the one with the greatest $r - l$.**

Example

stdin	stdout

```
10
15
16
2
3
9699690
223092870
847288609
900660121
987698769
999999999
```
	```
	1 5
	16 16
	2 2
	1 2
	16 4404
	129 21123
	4112949 4113154
	15006 45016
	46887 64604
	163837 169830
	```