BIUPC-2025 Spring Preliminary Mock Contest(Junior division).

A. Twins

2 seconds, 256 megabytes

Imagine that you have a twin brother or sister. Having another person that looks exactly like you seems very unusual. It's hard to say if having something of an alter ego is good or bad. And if you do have a twin, then you very well know what it's like.

Now let's imagine a typical morning in your family. You haven't woken up yet, and Mom is already going to work. She has been so hasty that she has nearly forgotten to leave the two of her darling children some money to buy lunches in the school cafeteria. She fished in the purse and found some number of coins, or to be exact, n coins of arbitrary values $a_1, a_2, ..., a_n$. But as Mom was running out of time, she didn't split the coins for you two. So she scribbled a note asking you to split the money equally.

As you woke up, you found Mom's coins and read her note. "But why split the money equally?" — you thought. After all, your twin is sleeping and he won't know anything. So you decided to act like that: pick for yourself some subset of coins so that the sum of values of your coins is **strictly larger** than the sum of values of the remaining coins that your twin will have. However, you correctly thought that if you take too many coins, the twin will suspect the deception. So, you've decided to stick to the following strategy to avoid suspicions: you take the **minimum number of coins**, whose sum of values is strictly more than the sum of values of the remaining coins. On this basis, determine what **minimum** number of coins you need to take to divide them in the described manner.

Input

The first line contains integer n $(1 \le n \le 100)$ — the number of coins. The second line contains a sequence of n integers $a_1, a_2, ..., a_n$ $(1 \le a_i \le 100)$ — the coins' values. All numbers are separated with spaces.

Output

In the single line print the single number — the minimum needed number of coins.

input
```
2
3 3
```
output
```
2
```

input
```
3
2 1 2
```
output
```
2
```

In the first sample you will have to take 2 coins (you and your twin have sums equal to $6, 0$ correspondingly). If you take 1 coin, you get sums $3, 3$. If you take 0 coins, you get sums $0, 6$. Those variants do not satisfy you as your sum should be strictly more that your twins' sum.

In the second sample one coin isn't enough for us, too. You can pick coins with values $1, 2$ or $2, 2$. In any case, the minimum number of coins equals 2.

B. Maximal Continuous Rest

2 seconds, 256 megabytes

Each day in Berland consists of n hours. Polycarp likes time management. That's why he has a fixed schedule for each day — it is a sequence $a_1, a_2, \ldots, a_n$ (each a_i is either 0 or 1), where $a_i = 0$ if Polycarp works during the i-th hour of the day and $a_i = 1$ if Polycarp rests during the i-th hour of the day.

Days go one after another endlessly and Polycarp uses the same schedule for each day.

What is the maximal number of continuous hours during which Polycarp rests? It is guaranteed that there is at least one working hour in a day.

Input

The first line contains n ($1 \le n \le 2 \cdot 10^5$) — number of hours per day.

The second line contains n integer numbers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 1$), where $a_i = 0$ if the i-th hour in a day is working and $a_i = 1$ if the i-th hour is resting. It is guaranteed that $a_i = 0$ for at least one i.

Output

Print the maximal number of continuous hours during which Polycarp rests. Remember that you should consider that days go one after another endlessly and Polycarp uses the same schedule for each day.

input
```
5
1 0 1 0 1
```
output
```
2
```

input
```
6
0 1 0 1 1 0
```
output
```
2
```

input
```
7
1 0 1 1 1 0 1
```
output
```
3
```

input
```
3
0 0 0
```
output
```
0
```

In the first example, the maximal rest starts in last hour and goes to the first hour of the next day.

In the second example, Polycarp has maximal rest from the 4-th to the 5-th hour.

In the third example, Polycarp has maximal rest from the 3-rd to the 5-th hour.

In the fourth example, Polycarp has no rest at all.

C. Candies

1 second, 256 megabytes

Polycarpus has got n candies and m friends ($n \geq m$). He wants to make a New Year present with candies to each friend. Polycarpus is planning to present all candies and he wants to do this in the fairest (that is, most equal) manner. He wants to choose such a_i, where a_i is the number of candies in the i-th friend's present, that the maximum a_i differs from the least a_i as little as possible.

For example, if n is divisible by m, then he is going to present the same number of candies to all his friends, that is, the maximum a_i won't differ from the minimum one.

Input

The single line of the input contains a pair of space-separated positive integers n, m ($1 \leq n, m \leq 100; n \geq m$) — the number of candies and the number of Polycarpus's friends.

Output

Print the required sequence $a_1, a_2, ..., a_m$, where a_i is the number of candies in the i-th friend's present. All numbers a_i must be positive integers, total up to n, the maximum one should differ from the minimum one by the smallest possible value.

input
```
12 3
```
output
```
4 4 4
```

input
```
15 4
```
output
```
3 4 4 4
```

input
```
18 7
```
output
```
2 2 2 3 3 3 3
```

Print a_i in any order, separate the numbers by spaces.

D. New Year and Naming

1 second, 1024 megabytes

Happy new year! The year 2020 is also known as *Year Gyeongja* (경자년, *gyeongja-nyeon*) in Korea. Where did the name come from? Let's briefly look at the *Gapja* system, which is traditionally used in Korea to name the years.

There are two sequences of n strings $s_1, s_2, s_3, \ldots, s_n$ and m strings $t_1, t_2, t_3, \ldots, t_m$. These strings contain only lowercase letters. There might be duplicates among these strings.

Let's call a concatenation of strings x and y as the string that is obtained by writing down strings x and y one right after another without changing the order. For example, the concatenation of the strings "code" and "forces" is the string "codeforces".

The year 1 has a name which is the concatenation of the two strings s_1 and t_1. When the year increases by one, we concatenate the next two strings in order from each of the respective sequences. If the string that is currently being used is at the end of its sequence, we go back to the first string in that sequence.

For example, if $n = 3, m = 4, s =${"a", "b", "c"}, $t =${"d", "e", "f", "g"}, the following table denotes the resulting year names. Note that the names of the years may repeat.

Year	S	T	Name	Year	S	T	Name
1	a	d	ad	8	b	g	bg
2	b	e	be	9	c	d	cd
3	c	f	cf	10	a	e	ae
4	a	g	ag	11	b	f	bf
5	b	d	bd	12	c	g	cg
6	c	e	ce	13	a	d	ad
7	a	f	af	14	b	e	be

You are given two sequences of strings of size n and m and also q queries. For each query, you will be given the current year. Could you find the name corresponding to the given year, according to the *Gapja* system?

Input

The first line contains two integers n, m ($1 \leq n, m \leq 20$).

The next line contains n strings $s_1, s_2, \ldots, s_n$. Each string contains only lowercase letters, and they are separated by spaces. The length of each string is at least 1 and at most 10.

The next line contains m strings $t_1, t_2, \ldots, t_m$. Each string contains only lowercase letters, and they are separated by spaces. The length of each string is at least 1 and at most 10.

Among the given $n + m$ strings may be duplicates (that is, they are not necessarily all different).

The next line contains a single integer q ($1 \leq q \leq 2\,020$).

In the next q lines, an integer y ($1 \leq y \leq 10^9$) is given, denoting the year we want to know the name for.

Output

Print q lines. For each line, print the name of the year as per the rule described above.

input
```
10 12
sin im gye gap eul byeong jeong mu gi gyeong
yu sul hae ja chuk in myo jin sa o mi sin
14
1
2
3
4
10
11
12
13
73
2016
2017
2018
2019
2020
```

output

```
sinyu
imsul
gyehae
gapja
gyeongo
sinmi
imsin
gyeyu
gyeyu
byeongsin
jeongyu
musul
gihae
gyeongja
```

The first example denotes the actual names used in the *Gapja* system. These strings usually are either a number or the name of some animal.

E. Good Kid

1 second, 256 megabytes

Slavic is preparing a present for a friend's birthday. He has an array a of n digits and the present will be the product of all these digits. Because Slavic is a good kid who wants to make the biggest product possible, he wants to add 1 to exactly one of his digits.

What is the maximum product Slavic can make?

Input

The first line contains a single integer t ($1 \leq t \leq 10^4$) — the number of test cases.

The first line of each test case contains a single integer n ($1 \leq n \leq 9$) — the number of digits.

The second line of each test case contains n space-separated integers a_i ($0 \leq a_i \leq 9$) — the digits in the array.

Output

For each test case, output a single integer — the maximum product Slavic can make, by adding 1 to exactly one of his digits.

input

```
4
4
2 2 1 2
3
0 1 2
5
4 3 2 3 4
9
9 9 9 9 9 9 9 9 9
```

output

```
16
2
432
430467210
```

F. Nicholas and Permutation

1 second, 256 megabytes

Nicholas has an array a that contains n **distinct** integers from 1 to n. In other words, Nicholas has a permutation of size n.

Nicholas want the minimum element (integer 1) and the maximum element (integer n) to be as far as possible from each other. He wants to perform exactly one swap in order to maximize the distance between the minimum and the maximum elements. The distance between two elements is considered to be equal to the absolute difference between their positions.

Input

The first line of the input contains a single integer n $(2 \leq n \leq 100)$ — the size of the permutation.

The second line of the input contains n distinct integers $a_1, a_2, ..., a_n$ $(1 \leq a_i \leq n)$, where a_i is equal to the element at the i-th position.

Output

Print a single integer — the maximum possible distance between the minimum and the maximum elements Nicholas can achieve by performing exactly one swap.

input

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

output

```
3
```

input

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

output

```
6
```

input

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

output

```
5
```

In the first sample, one may obtain the optimal answer by swapping elements 1 and 2.

In the second sample, the minimum and the maximum elements will be located in the opposite ends of the array if we swap 7 and 2.

In the third sample, the distance between the minimum and the maximum elements is already maximum possible, so we just perform some unnecessary swap, for example, one can swap 5 and 2.

G. Binary Decimal

1 second, 512 megabytes

Let's call a number a *binary decimal* if it's a positive integer and all digits in its decimal notation are either 0 or 1. For example, $1\,010\,111$ is a binary decimal, while $10\,201$ and $787\,788$ are not.

Given a number n, you are asked to represent n as a sum of some (not necessarily distinct) binary decimals. Compute the smallest number of binary decimals required for that.

Input

The first line contains a single integer t $(1 \leq t \leq 1000)$, denoting the number of test cases.

The only line of each test case contains a single integer n ($1 \leq n \leq 10^9$), denoting the number to be represented.

Output

For each test case, output the smallest number of binary decimals required to represent n as a sum.

input

```
3
121
5
1000000000
```

output

```
2
5
1
```

In the first test case, 121 can be represented as $121 = 110 + 11$ or $121 = 111 + 10$.

In the second test case, 5 can be represented as $5 = 1 + 1 + 1 + 1 + 1$.

In the third test case, $1\,000\,000\,000$ is a binary decimal itself, thus the answer is 1.

H. Dinner Time

1 second, 256 megabytes

Given four integers n, m, p, and q, determine whether there exists an integer array $a_1, a_2, \ldots, a_n$ (elements may be negative) satisfying the following conditions:

- The sum of all elements in the array is equal to m:

$$a_1 + a_2 + \ldots + a_n = m$$

- The sum of every p consecutive elements is equal to q:

$$a_i + a_{i+1} + \ldots + a_{i+p-1} = q, \qquad \text{for all } 1 \le i \le n - p + 1$$

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^4$). The description of the test cases follows.

The first and only line of each test case contains four integers n, m, p, and q ($1 \le p \le n \le 100$, $1 \le q, m \le 100$) — the length of the array, the sum of elements, the length of a segment, and the sum of a segment, respectively.

Output

For each test case, output "YES" (without quotes) if there exists an array satisfying the above conditions, and "NO" (without quotes) otherwise.

You can output "YES" and "NO" in any case (for example, strings "yES", "yes", and "Yes" will all be recognized as valid responses).

input

```
5
3 2 2 1
1 1 1 1
5 4 2 3
10 7 5 2
4 4 1 3
```

output

```
YES
YES
YES
NO
NO
```

In the first test case, an example of an array satisfying the condition is $[1, 0, 1]$. This is because:

- $a_1 + a_2 + a_3 = 1 + 0 + 1 = 2 = m$
- $a_1 + a_2 = 1 + 0 = 1 = q$
- $a_2 + a_3 = 0 + 1 = 1 = q$

In the second test case, the only array satisfying the condition is $[1]$.

In the third test case, an example of an array satisfying the condition is $[-2, 5, -2, 5, -2]$.

In the fourth test case, it can be proven that there is no array satisfying the condition.

I. Twice

1 second, 256 megabytes

Kinich wakes up to the start of a new day. He turns on his phone, checks his mailbox, and finds a mysterious present. He decides to unbox the present.

Kinich unboxes an array a with n integers. Initially, Kinich's score is 0. He will perform the following operation any number of times:

- Select two indices i and j $(1 \le i < j \le n)$ such that neither i nor j has been chosen in any previous operation and $a_i = a_j$. Then, add 1 to his score.

Output the maximum score Kinich can achieve after performing the aforementioned operation any number of times.

Input

The first line contains an integer t $(1 \le t \le 500)$ — the number of test cases.

The first line of each test case contains an integer n $(1 \le n \le 20)$ — the length of a.

The following line of each test case contains n space-separated integers $a_1, a_2, \ldots, a_n$ $(1 \le a_i \le n)$.

Output

For each test case, output the maximum score achievable on a new line.

input

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

output

```
0
1
0
1
3
```

In the first and third testcases, Kinich cannot perform any operations.

In the second testcase, Kinich can perform one operation with $i = 1$ and $j = 2$.

In the fourth testcase, Kinich can perform one operation with $i = 1$ and $j = 4$.

J. Squares and Cubes

1 second, 256 megabytes

Polycarp likes squares and cubes of positive integers. Here is the beginning of the sequence of numbers he likes: $1, 4, 8, 9, \ldots$.

For a given number n, count the number of integers from 1 to n that Polycarp likes. In other words, find the number of such x that x is a square of a positive integer number or a cube of a positive integer number (or both a square and a cube simultaneously).

Input

The first line contains an integer t $(1 \le t \le 20)$ — the number of test cases.

Then t lines contain the test cases, one per line. Each of the lines contains one integer n $(1 \le n \le 10^9)$.

Output

For each test case, print the answer you are looking for — the number of integers from 1 to n that Polycarp likes.

input

```
6
10
1
25
1000000000
999999999
500000000
```

output

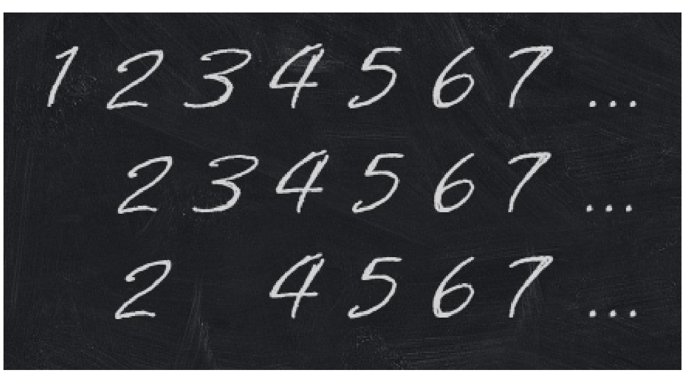
```
4
1
6
32591
32590
23125
```

K. Remove a Progression

2 seconds, 256 megabytes

You have a list of numbers from 1 to n written from left to right on the blackboard.

You perform an algorithm consisting of several steps (steps are 1-indexed). On the i-th step you wipe the i-th number (considering only **remaining** numbers). You wipe *the whole number* (not one digit).



When there are less than i numbers remaining, you stop your algorithm.

Now you wonder: what is the value of the x-th remaining number after the algorithm is stopped?

Input

The first line contains one integer T $(1 \le T \le 100)$ — the number of queries. The next T lines contain queries — one per line. All queries are independent.

Each line contains two space-separated integers n and x ($1 \le x < n \le 10^9$) — the length of the list and the position we wonder about. It's guaranteed that after the algorithm ends, the list will still contain at least x numbers.

Output

Print T integers (one per query) — the values of the x-th number after performing the algorithm for the corresponding queries.

input

```
3
3 1
4 2
69 6
```

output

```
2
4
12
```

L. Long Comparison

2 seconds, 256 megabytes

Monocarp wrote down two numbers on a whiteboard. Both numbers follow a specific format: a positive integer x with p zeros appended to its end.

Now Monocarp asks you to compare these two numbers. Can you help him?

Input

The first line contains a single integer t $(1 \le t \le 10^4)$ — the number of testcases.

The first line of each testcase contains two integers x_1 and p_1 ($1 \le x_1 \le 10^6; 0 \le p_1 \le 10^6$) — the description of the first number.

The second line of each testcase contains two integers x_2 and p_2 ($1 \le x_2 \le 10^6; 0 \le p_2 \le 10^6$) — the description of the second number.

Output

For each testcase print the result of the comparison of the given two numbers. If the first number is smaller than the second one, print '<'. If the first number is greater than the second one, print '>'. If they are equal, print '='.

input

```
5
2 1
19 0
10 2
100 1
1999 0
2 3
1 0
1 0
99 0
1 2
```

output

```
>
=
<
=
<
```

The comparisons in the example are: $20 > 19$, $1000 = 1000$, $1999 < 2000$, $1 = 1$, $99 < 100$.

M. Copy-paste

1 second, 256 megabytes

— Hey folks, how do you like this problem?
— That'll do it.

BThero is a powerful magician. He has got n piles of candies, the i-th pile initially contains a_i candies. *BThero* can cast a *copy-paste* spell as follows:

1. He chooses two piles (i, j) such that $1 \le i, j \le n$ and $i \ne j$.
2. All candies from pile i are copied into pile j. Formally, the operation $a_j := a_j + a_i$ is performed.

BThero can cast this spell any number of times he wants to — but unfortunately, if some pile contains strictly more than k candies, he loses his magic power. What is the maximum number of times *BThero* can cast the spell without losing his power?

Input

The first line contains one integer T $(1 \le T \le 500)$ — the number of test cases.

Each test case consists of two lines:

- the first line contains two integers n and k ($2 \le n \le 1000$, $2 \le k \le 10^4$);
- the second line contains n integers a_1, a_2, ..., a_n ($1 \le a_i \le k$).

It is guaranteed that the sum of n over all test cases does not exceed 1000, and the sum of k over all test cases does not exceed 10^4.

Output

For each test case, print one integer — the maximum number of times *BThero* can cast the spell without losing his magic power.

input

```
3
2 2
1 1
3 5
1 2 3
3 7
3 2 2
```

output

```
1
5
4
```

In the first test case we get either $a = [1, 2]$ or $a = [2, 1]$ after casting the spell for the first time, and it is impossible to cast it again.

N. Scale

2 seconds, 256 megabytes

Tina has a square grid with n rows and n columns. Each cell in the grid is either 0 or 1.

Tina wants to reduce the grid by a factor of k (k **is a divisor of** n). To do this, Tina splits the grid into $k \times k$ nonoverlapping blocks of cells such that every cell belongs to exactly one block.

Tina then replaces each block of cells with a single cell equal to the value of the cells in the block. **It is guaranteed that every cell in the same block has the same value.**

For example, the following demonstration shows a grid being reduced by factor of 3.

Original grid

0	0	0	1	1	1
0	0	0	1	1	1
0	0	0	1	1	1
1	1	1	0	0	0
1	1	1	0	0	0
1	1	1	0	0	0

Reduced grid

0	1
1	0

Help Tina reduce the grid by a factor of k.

Input

The first line contains t $(1 \le t \le 100)$ – the number of test cases.

The first line of each test case contains two integers n and k ($1 \le n \le 1000$, $1 \le k \le n$, k **is a divisor of** n) — the number of rows and columns of the grid, and the factor that Tina wants to reduce the grid by.

Each of the following n lines contain n characters describing the cells of the grid. Each character is either 0 or 1. It is guaranteed every k by k block has the same value.

It is guaranteed the sum of n over all test cases does not exceed 1000.

Output

For each test case, output the grid reduced by a factor of k on a new line.

input

```
4
4 4
0000
0000
0000
0000
6 3
000111
000111
000111
111000
111000
111000
6 2
001100
001100
111111
111111
110000
110000
8 1
11111111
11111111
11111111
11111111
11111111
11111111
11111111
11111111
```

output

```
0
01
10
010
111
100
11111111
11111111
11111111
11111111
11111111
11111111
11111111
11111111
```