Codeforces Round 957 (Div. 3)

A. Only Pluses

1 second, 256 megabytes

Kmes has written three integers a, b and c in order to remember that he has to give Noobish_Monk $a \times b \times c$ bananas.

Noobish_Monk has found these integers and decided to do the following **at most** 5 times:

- pick one of these integers;
- increase it by 1.

For example, if $a = 2$, $b = 3$ and $c = 4$, then one can increase a three times by one and increase b two times. After that $a = 5$, $b = 5$, $c = 4$. Then the total number of bananas will be $5 \times 5 \times 4 = 100$.

What is the maximum value of $a \times b \times c$ Noobish_Monk can achieve with these operations?

Input

Each test contains multiple test cases. The first line of input contains a single integer t ($1 \le t \le 1000$) — the number of test cases. The description of the test cases follows.

The first and only line of each test case contains three integers a, b and c ($1 \le a, b, c \le 10$) — Kmes's integers.

Output

For each test case, output a single integer — the maximum amount of bananas Noobish_Monk can get.

input
2 2 3 4 10 1 10

output
100 600

B. Angry Monk

2 seconds, 256 megabytes

To celebrate his recovery, k1o0n has baked an enormous n metres long potato casserole.

Turns out, Noobish_Monk just can't stand potatoes, so he decided to ruin k1o0n's meal. He has cut it into k pieces, of lengths $a_1, a_2, \ldots, a_k$ meters.

k1o0n wasn't keen on that. Luckily, everything can be fixed. In order to do that, k1o0n can do one of the following operations:

- Pick a piece with length $a_i \ge 2$ and divide it into two pieces with lengths 1 and $a_i - 1$. As a result, the number of pieces will increase by 1;
- Pick a slice a_i and another slice with length $a_j = 1$ ($i \ne j$) and merge them into one piece with length $a_i + 1$. As a result, the number of pieces will decrease by 1.

Help k1o0n to find the minimum number of operations he needs to do in order to merge the casserole into one piece with length n.

For example, if $n = 5$, $k = 2$ and $a = [3, 2]$, it is optimal to do the following:

1. Divide the piece with length 2 into two pieces with lengths $2 - 1 = 1$ and 1, as a result $a = [3, 1, 1]$.
2. Merge the piece with length 3 and the piece with length 1, as a result $a = [4, 1]$.
3. Merge the piece with length 4 and the piece with length 1, as a result $a = [5]$.

Input

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

Description of each test case consists of two lines. The first line contains two integers n and k ($2 \le n \le 10^9$, $2 \le k \le 10^5$) — length of casserole and the number of pieces.

The second line contains k integers $a_1, a_2, \ldots, a_k$ ($1 \le a_i \le n - 1$, $\sum a_i = n$) — lengths of pieces of casserole, which Noobish_Monk has cut.

It is guaranteed that the sum of k over all t test cases doesn't exceed $2 \cdot 10^5$.

Output

For each test case, output the minimum number of operations K1o0n needs to restore his pie after the terror of Noobish_Monk.

input
4 5 3 3 1 1 5 2 3 2 11 4 2 3 1 5 16 6 1 6 1 1 1 6

output
2 3 9 15

C. Gorilla and Permutation

2 seconds, 256 megabytes

Gorilla and Noobish_Monk found three numbers n, m, and k ($m < k$). They decided to construct a permutation [†] of length n.

For the permutation, Noobish_Monk came up with the following function: $g(i)$ is the sum of all the numbers in the permutation on a prefix of length i that are not greater than m. Similarly, Gorilla came up with the function f, where $f(i)$ is the sum of all the numbers in the permutation on a prefix of length i that are not less than k. A prefix of length i is a subarray consisting of the first i elements of the original array.

For example, if $n = 5$, $m = 2$, $k = 5$, and the permutation is $[5, 3, 4, 1, 2]$, then:

- $f(1) = 5$, because $5 \ge 5$; $g(1) = 0$, because $5 > 2$;
- $f(2) = 5$, because $3 < 5$; $g(2) = 0$, because $3 > 2$;
- $f(3) = 5$, because $4 < 5$; $g(3) = 0$, because $4 > 2$;
- $f(4) = 5$, because $1 < 5$; $g(4) = 1$, because $1 \le 2$;
- $f(5) = 5$, because $2 < 5$; $g(5) = 1 + 2 = 3$, because $2 \le 2$.

Help them find a permutation for which the value of $\left(\sum_{i=1}^{n} f(i) - \sum_{i=1}^{n} g(i)\right)$ is maximized.

†A permutation of length n is an array consisting of n distinct integers from 1 to n in any order. For example, $[2, 3, 1, 5, 4]$ is a permutation, but $[1, 2, 2]$ is not a permutation (as 2 appears twice in the array) and $[1, 3, 4]$ is also not a permutation (as $n = 3$, but 4 appears in the array).

Input

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

The only line of each case contains three integers n, m, k ($2 \le n \le 10^5$; $1 \le m < k \le n$) — the size of the permutation to be constructed and two integers.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output the permutation — a set of numbers that satisfies the conditions of the problem. If there are multiple solutions, output any of them.

input
3 5 2 5 3 1 3 10 3 8

output
5 3 4 1 2 3 2 1 10 9 8 4 7 5 6 1 2 3

In the first example,

$$\left(\sum_{i=1}^{n} f(i) - \sum_{i=1}^{n} g(i)\right) = 5 \cdot 5 - (0 \cdot 3 + 1 + 3) = 25 - 4 = 21$$

D. Test of Love

2 seconds, 256 megabytes

ErnKor is ready to do anything for Julen, even to swim through crocodile-infested swamps. We decided to test this love. ErnKor will have to swim across a river with a width of 1 meter and a length of n meters.

The river is very cold. Therefore, **in total** (that is, throughout the entire swim from 0 to $n + 1$) ErnKor can swim in the water for no more than k meters. For the sake of humanity, we have added not only crocodiles to the river, but also logs on which he can jump. Our test is as follows:

Initially, ErnKor is on the left bank and needs to reach the right bank. They are located at the 0 and $n + 1$ meters respectively. The river can be represented as n *segments*, each with a length of 1 meter. Each *segment* contains either a log `'L'`, a crocodile `'C'`, or just water `'W'`. ErnKor can move as follows:

- If he is on the surface (i.e., on the bank or on a log), he can jump forward for no more than m ($1 \le m \le 10$) meters (he can jump on the bank, on a log, or in the water).
- If he is in the water, he can only swim to the next river *segment* (or to the bank if he is at the n-th meter).
- ErnKor cannot land in a *segment* with a crocodile in any way.

Determine if ErnKor can reach the right bank.

Input

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

The first line of each test case contains three numbers n, m, k ($0 \le k \le 2 \cdot 10^5$, $1 \le n \le 2 \cdot 10^5$, $1 \le m \le 10$) — the length of the river, the distance ErnKor can jump, and the number of meters ErnKor can swim without freezing.

The second line of each test case contains a string a of length n. a_i denotes the object located at the i-th meter. ($a_i \in$ {`'W'`,`'C'`,`'L'`})

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each test case, output "`YES`" if ErnKor can pass the test, and output "`NO`" otherwise.

You can output the answer in any case (upper or lower). For example, the strings "`yEs`", "`yes`", "`Yes`", and "`YES`" will be recognized as positive responses.

input
6 6 2 0 LWLLLW 6 1 1 LWLLLL 6 1 1 LWLLWL 6 2 15 LWLLCC 6 10 0 CCCCCC 6 6 1 WCCCCW

output
YES YES NO NO YES YES

Let's consider examples:

- First example: We jump from the shore to the first log ($0 \to 1$), from the first log to the second ($1 \to 3$), from the second to the fourth ($3 \to 5$), and from the last log to the shore ($5 \to 7$). So, we have $0 \to 1 \to 3 \to 5 \to 7$. Since we did not encounter a crocodile and swam no more than k meters, the answer is «`YES`».
- Second example: $0 \to 1$, we jump into the water from the first log ($1 \to 2$), swim a cell to the log ($2 \rightsquigarrow 3$), $3 \to 4 \to 5 \to 6 \to 7$. Since we did not encounter a crocodile and swam no more than k meters, the answer is «`YES`».
- In the third example, ErnKor needs to swim two cells `'W'`, but can only swim one. Therefore, the answer is «`NO`».
- Sixth example: We jump from the shore into the water ($0 \to 6$) and swim one cell in the water ($6 \rightsquigarrow 7$). Since we did not encounter a crocodile and swam no more than k meters, the answer is «`YES`».

E. Novice's Mistake

3 seconds, 256 megabytes

One of the first programming problems by K1o0n looked like this: "Noobish_Monk has n $(1 \le n \le 100)$ friends. Each of them gave him a $(1 \le a \le 10000)$ apples for his birthday. Delighted with such a gift, Noobish_Monk returned b $(1 \le b \le \min(10000, a \cdot n))$ apples to his friends. How many apples are left with Noobish_Monk?"

K1o0n wrote a solution, but accidentally considered the value of n as a string, so the value of $n \cdot a - b$ was calculated differently. Specifically:

- when multiplying the string n by the integer a, he will get the string $s = \underbrace{n + n + \cdots + n + n}_{a \text{ times}}$
- when subtracting the integer b from the string s, the last b characters will be removed from it. If b is greater than or equal to the length of the string s, it will become empty.

Learning about this, ErnKor became interested in how many pairs (a, b) exist for a given n, satisfying the constraints of the problem, on which K1o0n's solution gives the correct answer.

"The solution gives the correct answer" means that it outputs a **non-empty** string, and this string, when converted to an integer, equals the correct answer, i.e., the value of $n \cdot a - b$.

Input
The first line contains a single integer t $(1 \le t \le 100)$ — the number of test cases.

For each test case, a single line of input contains an integer n $(1 \le n \le 100)$.

It is guaranteed that in all test cases, n is distinct.

Output
For each test case, output the answer in the following format:

In the first line, output the integer x — the number of bad tests for the given n.

In the next x lines, output two integers a_i and b_i — such integers that K1o0n's solution on the test "n a_i b_i" gives the correct answer.

input

```
3
2
3
10
```

output

```
3
20 18
219 216
2218 2214
1
165 162
1
1262 2519
```

In the first example, $a = 20$, $b = 18$ are suitable, as "2" $\cdot 20 - 18 =$ "22222222222222222222"$- 18 = 22 = 2 \cdot 20 - 18$

F. Valuable Cards

4 seconds, 512 megabytes

In his favorite cafe Kmes once again wanted to try the herring under a fur coat. Previously, it would not have been difficult for him to do this, but the cafe recently introduced a new purchasing policy.

Now, in order to make a purchase, Kmes needs to solve the following problem: n cards with prices for different positions are laid out in front of him, on the i-th card there is an integer a_i, among these prices there is no whole positive integer x.

Kmes is asked to divide these cards into the minimum number of *bad* segments (so that each card belongs to exactly one segment). A segment is considered *bad* if it is impossible to select a subset of cards with a product equal to x.

Formally, the segment (l, r) is *bad* if there are no indices $i_1 < i_2 < \ldots < i_k$ such that $l \le i_1, i_k \le r$, and $a_{i_1} \cdot a_{i_2} \ldots \cdot a_{i_k} = x$.

Help Kmes determine the minimum number of *bad* segments in order to enjoy his favorite dish.

Input
The first line contains a single integer t $(1 \le t \le 10^3)$ — the number of test cases.

The first line of each set of input data gives you 2 integers n and x ($1 \le n \le 10^5, 2 \le x \le 10^5$) — the number of cards and the integer, respectively.

The second line of each set of input data contains n integers a_i ($1 \le a_i \le 2 \cdot 10^5, a_i \ne x$) — the prices on the cards.

It is guaranteed that the sum of n over all sets of test data does not exceed 10^5.

Output
For each set of input data, output the minimum number of *bad* segments.

input

```
8
6 4
2 3 6 2 1 2
9 100000
50000 25000 12500 6250 3125 2 4 8 16
5 2
1 1 1 1 1
8 6
4 3 4 3 4 3 4 3
7 12
6 11 1 3 11 10 2
10 5
2 4 4 2 4 4 4 3 1 1
7 8
4 6 5 1 2 4 1
8 27
3 9 17 26 2 20 9 3
```

output

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

G. Ultra-Meow

2.5 seconds, 256 megabytes

K1o0n gave you an array a of length n, consisting of numbers $1, 2, \ldots, n$. Accept it? Of course! But what to do with it? Of course, calculate $\text{MEOW}(a)$.

Let $\text{MEX}(S, k)$ be the k-th **positive** (strictly greater than zero) integer in ascending order that is not present in the set S. Denote $\text{MEOW}(a)$ as the sum of $\text{MEX}(b, |b| + 1)$, over all **distinct** subsets b of the array a.

Examples of $\text{MEX}(S, k)$ values for sets:

- $\text{MEX}(\{3, 2\}, 1) = 1$, because 1 is the first positive integer not present in the set;
- $\text{MEX}(\{4, 2, 1\}, 2) = 5$, because the first two positive integers not present in the set are 3 and 5;
- $\text{MEX}(\{\}, 4) = 4$, because there are no numbers in the empty set, so the first 4 positive integers not present in it are $1, 2, 3, 4$.

Input

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

In a single line of each test case, an integer n ($1 \le n \le 5000$) is entered, the size of the array of gifted numbers.

It is guaranteed that the sum of n^2 over all test cases does not exceed $25 \cdot 10^6$.

Output

For each test case, output a single number — $\text{MEOW}(a)$. Since it may be very large, output it modulo $10^9 + 7$.

input
5 2 3 4999 5 1

output
12 31 354226409 184 4