Codeforces Round #678 (Div. 2)

A. Reorder

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

For a given array a consisting of n integers and a given integer m find if it is possible to reorder elements of the array a in such a way that $\sum_{i=1}^{n}\sum_{j=i}^{n}\frac{a_j}{j}$ equals m? It is forbidden to delete elements as well as insert new elements. Please note that no rounding occurs during division, for example, $\frac{5}{2}=2.5$.

Input

The first line contains a single integer t — the number of test cases ($1 \le t \le 100$). The test cases follow, each in two lines.

The first line of a test case contains two integers n and m ($1 \le n \le 100$, $0 \le m \le 10^6$). The second line contains integers $a_1, a_2, \ldots, a_n$ ($0 \le a_i \le 10^6$) — the elements of the array.

Output

For each test case print "YES", if it is possible to reorder the elements of the array in such a way that the given formula gives the given value, and "NO" otherwise.

Example

input

```
2
3 8
2 5 1
4 4
0 1 2 3
```

output

```
YES
NO
```

Note

In the first test case one of the reorders could be $[1, 2, 5]$. The sum is equal to $(\frac{1}{1}+\frac{2}{2}+\frac{5}{3})+(\frac{2}{2}+\frac{5}{3})+(\frac{5}{3})=8$. The brackets denote the inner sum $\sum_{j=i}^{n}\frac{a_j}{j}$, while the summation of brackets corresponds to the sum over i.

B. Prime Square

time limit per test: 1.5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Sasha likes investigating different math objects, for example, magic squares. But Sasha understands that magic squares have already been studied by hundreds of people, so he sees no sense of studying them further. Instead, he invented his own type of square — a prime square.

A square of size $n \times n$ is called prime if the following three conditions are held simultaneously:

- all numbers on the square are non-negative integers not exceeding 10^5;
- there are no prime numbers in the square;
- sums of integers in each row and each column are prime numbers.

Sasha has an integer n. He asks you to find any prime square of size $n \times n$. Sasha is absolutely sure such squares exist, so just help him!

Input

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

Each of the next t lines contains a single integer n ($2 \le n \le 100$) — the required size of a square.

Output

For each test case print n lines, each containing n integers — the prime square you built. If there are multiple answers, print any.

Example

input

```
2
4
2
```

output

```
4 6 8 1
4 9 9 9
4 10 10 65
1 4 4 4
1 1
1 1
```

C. Binary Search

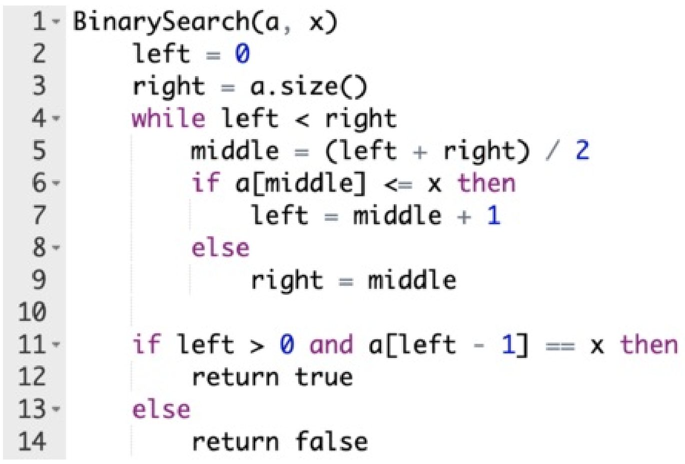
time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

Andrey thinks he is truly a successful developer, but in reality he didn't know about the binary search algorithm until recently. After reading some literature Andrey understood that this algorithm allows to quickly find a certain number x in an array. For an array a indexed from zero, and an integer x the pseudocode of the algorithm is as follows:



```
BinarySearch(a, x)
    left = 0
    right = a.size()
    while left < right
        middle = (left + right) / 2
        if a[middle] <= x then
            left = middle + 1
        else
            right = middle

    if left > 0 and a[left - 1] == x then
        return true
    else
        return false
```

Note that the elements of the array are indexed from zero, and the division is done in integers (rounding down).

Andrey read that the algorithm only works if the array is sorted. However, he found this statement untrue, because there certainly exist unsorted arrays for which the algorithm find x!

Andrey wants to write a letter to the book authors, but before doing that he must consider the permutations of size n such that the algorithm finds x in them. A permutation of size n is an array consisting of n distinct integers between 1 and n in arbitrary order.

Help Andrey and find the number of permutations of size n which contain x at position pos and for which the given implementation of the binary search algorithm finds x (returns true). As the result may be extremely large, print the remainder of its division by $10^9 + 7$.

Input

The only line of input contains integers n, x and pos ($1 \le x \le n \le 1000$, $0 \le pos \le n - 1$) — the required length of the permutation, the number to search, and the required position of that number, respectively.

Output

Print a single number — the remainder of the division of the number of valid permutations by $10^9 + 7$.

Examples

input

```
4 1 2
```

output

```
6
```

input

```
123 42 24
```

output

```
824071958
```

Note

All possible permutations in the first test case: $(2, 3, 1, 4)$, $(2, 4, 1, 3)$, $(3, 2, 1, 4)$, $(3, 4, 1, 2)$, $(4, 2, 1, 3)$, $(4, 3, 1, 2)$.

D. Bandit in a City

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

Bandits appeared in the city! One of them is trying to catch as many citizens as he can.

The city consists of n squares connected by $n-1$ roads in such a way that it is possible to reach any square from any other square. The square number 1 is the main square.

After Sunday walk all the roads were changed to **one-way** roads in such a way that it is possible to reach any square from the main square.

At the moment when the bandit appeared on the main square there were a_i citizens on the i-th square. Now the following process will begin. First, each citizen that is currently on a square with some outgoing one-way roads chooses one of such roads and moves along it to another square. Then the bandit chooses one of the one-way roads outgoing from the square he is located and moves along it. The process is repeated until the bandit is located on a square with no outgoing roads. The bandit catches all the citizens on that square.

The bandit wants to catch as many citizens as possible; the citizens want to minimize the number of caught people. The bandit and the citizens know positions of all citizens at any time, the citizens can cooperate. If both sides act optimally, how many citizens will be caught?

Input

The first line contains a single integer n — the number of squares in the city ($2 \le n \le 2 \cdot 10^5$).

The second line contains $n-1$ integers $p_2, p_3 \dots p_n$ meaning that there is a one-way road from the square p_i to the square i ($1 \le p_i < i$).

The third line contains n integers $a_1, a_2, \dots, a_n$ — the number of citizens on each square initially ($0 \le a_i \le 10^9$).

Output

Print a single integer — the number of citizens the bandit will catch if both sides act optimally.

Examples

input

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

output

```
3
```

input

```
3
1 1
3 1 3
```

output

```
4
```

Note

In the first example the citizens on the square 1 can split into two groups $2+1$, so that the second and on the third squares will have 3 citizens each.

In the second example no matter how citizens act the bandit can catch at least 4 citizens.

E. Complicated Computations

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

In this problem MEX of a certain array is the smallest **positive** integer not contained in this array.

Everyone knows this definition, including Lesha. But Lesha loves MEX, so he comes up with a new problem involving MEX every day, including today.

You are given an array a of length n. Lesha considers all the **non-empty** subarrays of the initial array and computes MEX for each of them. Then Lesha computes MEX of the obtained numbers.

An array b is a subarray of an array a, if b can be obtained from a by deletion of several (possible none or all) elements from the beginning and several (possibly none or all) elements from the end. In particular, an array is a subarray of itself.

Lesha understands that the problem is very interesting this time, but he doesn't know how to solve it. Help him and find the MEX of MEXes of all the subarrays!

Input

The first line contains a single integer n ($1 \le n \le 10^5$) — the length of the array.

The next line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) — the elements of the array.

Output

Print a single integer — the MEX of MEXes of all subarrays.

Examples

input

```
3
1 3 2
```

output

```
3
```

input

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

output

```
6
```

F. Sum Over Subsets

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

You are given a multiset S. Over all pairs of subsets A and B, such that:

- $B \subset A$;
- $|B| = |A| - 1$;
- greatest common divisor of all elements in A is equal to one;

find the sum of $\sum_{x \in A} x \cdot \sum_{x \in B} x$, modulo $998\,244\,353$.

Input

The first line contains one integer m ($1 \le m \le 10^5$): the number of different values in the multiset S.

Each of the next m lines contains two integers a_i, $freq_i$ ($1 \le a_i \le 10^5, 1 \le freq_i \le 10^9$). Element a_i appears in the multiset S $freq_i$ times. All a_i are different.

Output

Print the required sum, modulo $998\,244\,353$.

Examples

input

```
2
1 1
2 1
```

output

```
9
```

input

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

output

```
1207
```

input

```
1
1 5
```

output

```
560
```

Note

A multiset is a set where elements are allowed to coincide. $|X|$ is the cardinality of a set X, the number of elements in it.

$A \subset B$: Set A is a subset of a set B.

In the first example $B = \{1\}, A = \{1, 2\}$ and $B = \{2\}, A = \{1, 2\}$ have a product equal to $1 \cdot 3 + 2 \cdot 3 = 9$. Other pairs of A and B don't satisfy the given constraints.