A. Split the Multiset

```
Input file:     standard input
Output file:    standard output
Time limit:     1 second
Memory limit:   512 megabytes
```

A *multiset* is a set of numbers in which there can be equal elements, and the order of the numbers does not matter. For example, $\{2, 2, 4\}$ is a multiset.

You have a multiset S. Initially, the multiset contains only one positive integer n. That is, $S = \{n\}$. Additionally, there is a given positive integer k.

In one operation, you can select any positive integer u in S and remove one copy of u from S. Then, insert no more than k positive integers into S so that the sum of all inserted integers is equal to u.

Find the minimum number of operations to make S contain n ones.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 1000$). Description of the test cases follows.

The only line of each testcase contains two integers n, k ($1 \le n \le 1000, 2 \le k \le 1000$).

Output

For each testcase, print one integer, which is the required answer.

Standard Input	Standard Output

4
1 5
5 2
6 3
16 4
	0
	4
	3
	5

Note

For the first test case, initially $S = \{1\}$, already satisfying the requirement. Therefore, we need zero operations.

For the second test case, initially $S = \{5\}$. We can apply the following operations:

- Select $u = 5$, remove u from S, and insert $2, 3$ into S. Now, $S = \{2, 3\}$.
- Select $u = 2$, remove u from S, and insert $1, 1$ into S. Now, $S = \{1, 1, 3\}$.
- Select $u = 3$, remove u from S, and insert $1, 2$ into S. Now, $S = \{1, 1, 1, 2\}$.
- Select $u = 2$, remove u from S, and insert $1, 1$ into S. Now, $S = \{1, 1, 1, 1, 1\}$.

Using 4 operations in total, we achieve the goal.

For the third test case, initially $S = \{6\}$. We can apply the following operations:

- Select $u = 6$, remove u from S, and insert $1, 2, 3$ into S. Now, $S = \{1, 2, 3\}$.
- Select $u = 2$, remove u from S, and insert $1, 1$ into S. Now, $S = \{1, 1, 1, 3\}$.
- Select $u = 3$, remove u from S, and insert $1, 1, 1$ into S. Now, $S = \{1, 1, 1, 1, 1, 1\}$.

Using 3 operations in total, we achieve the goal.

For the fourth test case, initially $S=\{16\}$. We can apply the following operations:

- Select $u=16$, remove u from S, and insert $4,4,4,4$ into S. Now, $S=\{4,4,4,4\}$.
- Repeat for 4 times: select $u=4$, remove u from S, and insert $1,1,1,1$ into S.

Using 5 operations in total, we achieve the goal.

,

B. Make Majority

```
Input file:     standard input
Output file:    standard output
Time limit:     1.5 seconds
Memory limit:   512 megabytes
```

You are given a sequence $[a_1, \ldots, a_n]$ where each element a_i is either 0 or 1. You can apply several (possibly zero) operations to the sequence. In each operation, you select two integers $1 \le l \le r \le |a|$ (where $|a|$ is the current length of a) and replace $[a_l, \ldots, a_r]$ with a single element x, where x is the majority of $[a_l, \ldots, a_r]$.

Here, the majority of a sequence consisting of 0 and 1 is defined as follows: suppose there are c_0 zeros and c_1 ones in the sequence, respectively.

- If $c_0 \ge c_1$, the majority is 0.
- If $c_0 < c_1$, the majority is 1.

For example, suppose $a = [1, 0, 0, 0, 1, 1]$. If we select $l = 1, r = 2$, the resulting sequence will be $[0, 0, 0, 1, 1]$. If we select $l = 4, r = 6$, the resulting sequence will be $[1, 0, 0, 1]$.

Determine if you can make $a = [1]$ with a finite number of operations.

Input

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

The first line of each testcase contains one integer n ($1 \le n \le 2 \cdot 10^5$).

The second line of each testcase contains a string consisting of 0 and 1, describing the sequence a.

It's guaranteed that the sum of n over all testcases does not exceed $2 \cdot 10^5$.

Output

For each testcase, if it's possible to make $a = [1]$, print YES. Otherwise, print NO. 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.

Standard Input	Standard Output
5 1 0 1 1 2 01 9 100000001 9 000011000	No Yes No Yes No

Note

In the fourth testcase of the example, initially $a = [1, 0, 0, 0, 0, 0, 0, 0, 1]$. A valid sequence of operations is:

1. Select $l = 2, r = 8$ and apply the operation. a becomes $[1, 0, 1]$.
2. Select $l = 1, r = 3$ and apply the operation. a becomes $[1]$.

,

C. Increasing Sequence with Fixed OR

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   512 megabytes
```

You are given a positive integer n. Find the **longest** sequence of positive integers $a = [a_1, a_2, \ldots, a_k]$ that satisfies the following conditions, and print the sequence:

- $a_i \le n$ for all $1 \le i \le k$.
- a is strictly increasing. That is, $a_i > a_{i-1}$ for all $2 \le i \le k$.
- $a_i \mid a_{i-1} = n$ for all $2 \le i \le k$, where $\mid$ denotes the bitwise OR operation.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 1000$). Description of the test cases follows.

The only line of each test case contains one integer n ($1 \le n \le 10^{18}$).

It's guaranteed that the sum of lengths of the longest valid sequences does not exceed $5 \cdot 10^5$.

Output

For each testcase, print two lines. In the first line, print the length of your constructed sequence, k. In the second line, print k positive integers, denoting the sequence. If there are multiple longest sequences, you can print any of them.

Standard Input	Standard Output

4
1
3
14
23
	1
	1
	3
	1 2 3
	4
	4 10 12 14
	5
	7 18 21 22 23

D. The Omnipotent Monster Killer

```
Input file:     standard input
Output file:    standard output
Time limit:     3 seconds
Memory limit:   512 megabytes
```

You, the monster killer, want to kill a group of monsters. The monsters are on a tree with n vertices. On vertex with number i ($1 \le i \le n$), there is a monster with a_i attack points. You want to battle with monsters for 10^{100} rounds.

In each round, the following happens in order:

1. All living monsters attack you. Your health decreases by the sum of attack points of all living monsters.
2. You select some (possibly all or none) monsters and kill them. After being killed, the monster will not be able to do any attacks in the future.

There is a restriction: in one round, you cannot kill two monsters that are directly connected by an edge.

If you choose what monsters to attack optimally, what is the smallest health decrement you can have after all rounds?

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 the test cases follows.

The first line of each test case contains an integer n ($1 \le n \le 3 \cdot 10^5$).

The second line of each test case contains n integers $a_1, \ldots, a_n$ ($1 \le a_i \le 10^{12}$).

The following $n-1$ lines each contain two integers x, y ($1 \le x, y \le n$), denoting an edge on the tree connecting vertex x and y.

It is guaranteed that the sum of n over all test cases does not exceed $3 \cdot 10^5$.

Output

For each test case, print one integer: the minimum possible health decrement.

Standard Input	Standard Output
3	1000000000000
1	193
1000000000000	57
5	
47 15 32 29 23	
1 2	
1 3	
2 4	
2 5	
7	
8 10 2 3 5 7 4	
1 2	
1 4	

3 2	
5 3	
6 2	
7 5	

Note

In the first test case, an optimal sequence of operations would be:

- In the first round: first, receive the attack from the monster on vertex 1, so your health decreases by 10^{12}. Then kill the monster on vertex 1.
- In the second round to the 10^{100}-th round: all monsters have been killed, so nothing happens.

The total health decrement is 10^{12}.

In the second test case, an optimal sequence of operations would be:

- In the first round: first, receive the attack from the monster on vertex $1,2,3,4,5$, so your health decreases by $47+15+32+29+23=146$. Then kill the monsters on vertex $1,4,5$.
- In the second round: first, receive the attack from the monster on vertex $2,3$, so your health decreases by $15+32=47$. Then kill the monsters on vertex $2,3$.
- In the third round to the 10^{100}-th round: all monsters have been killed, so nothing happens.

The total health decrement is 193.

In the third test case, an optimal sequence of operations would be:

- In the first round: first, receive the attack from the monster on vertex $1,2,3,4,5,6,7$, so your health decreases by $8+10+2+3+5+7+4=39$. Then kill the monsters on vertex $1,3,6,7$.
- In the second round: first, receive the attack from the monster on vertex $2,4,5$, so your health decreases by $10+3+5=18$. Then kill the monsters on vertex $2,4,5$.
- In the third round to the 10^{100}-th round: all monsters have been killed, so nothing happens.

The total health decrement is 57.

,

E. Range Minimum Sum

```
Input file:     standard input
Output file:    standard output
Time limit:     4 seconds
Memory limit:   512 megabytes
```

For an array $[a_1, a_2, \ldots, a_n]$ of length n, define $f(a)$ as the sum of the minimum element over all subsegments. That is,

$$f(a) = \sum_{l=1}^{n} \sum_{r=l}^{n} \min_{l \le i \le r} a_i.$$

A permutation is a sequence of integers from 1 to n of length n containing each number exactly once. You are given a permutation $[a_1, a_2, \ldots, a_n]$. For each i, solve the following problem independently:

- Erase a_i from a, concatenating the remaining parts, resulting in $b = [a_1, a_2, \ldots, a_{i-1},\ a_{i+1}, \ldots, a_n]$.
- Calculate $f(b)$.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 10^5$). Description of the test cases follows.

The first line of each test case contains an integer n ($1 \le n \le 5 \cdot 10^5$).

The second line of each test case contains n distinct integers $a_1, \ldots, a_n$ ($1 \le a_i \le n$).

It is guaranteed that the sum of n over all test cases does not exceed 10^6.

Output

For each test case, print one line containing n integers. The i-th integer should be the answer when erasing a_i.

Standard Input	Standard Output

4
1
1
3
3 1 2
5
4 2 1 5 3
8
8 1 4 6 7 3 5 2
	0
	4 7 5
	19 21 27 17 19
	79 100 72 68 67 80 73 80

Note

In the second test case, $a = [3, 1, 2]$.

- When removing a_1, $b = [1, 2]$. $f(b) = 1 + 2 + \min\{1, 2\} = 4$.
- When removing a_2, $b = [3, 2]$. $f(b) = 3 + 2 + \min\{3, 2\} = 7$.

- When removing a_3, $b = [3, 1]$. $f(b) = 3 + 1 + \min\{3, 1\} = 5$.

,

F. Heartbeat

```
Input file:     standard input
Output file:    standard output
Time limit:     5 seconds
Memory limit:   512 megabytes
```

For an array $u_1, u_2, \ldots, u_n$, define

- a prefix maximum as an index i such that $u_i > u_j$ for all $j < i$;
- a suffix maximum as an index i such that $u_i > u_j$ for all $j > i$;
- an ascent as an index i ($i > 1$) such that $u_i > u_{i-1}$.

You are given three cost arrays: $[a_1, a_2, \ldots, a_n]$, $[b_1, b_2, \ldots, b_n]$, and $[c_0, c_1, \ldots, c_{n-1}]$.

Define the *cost* of an array that has x prefix maximums, y suffix maximums, and z ascents as $a_x \cdot b_y \cdot c_z$.

Let the sum of costs of all permutations of $1, 2, \ldots, n$ be $f(n)$. Find $f(1)$, $f(2)$, ..., $f(n)$ modulo $998\,244\,353$.

Input

The first line contains an integer n ($1 \leq n \leq 700$).

The second line contains n integers $a_1, \ldots, a_n$ ($0 \leq a_i < 998\,244\,353$).

The third line contains n integers $b_1, \ldots, b_n$ ($0 \leq b_i < 998\,244\,353$).

The fourth line contains n integers $c_0, \ldots, c_{n-1}$ ($0 \leq c_i < 998\,244\,353$).

Output

Print n integers: the i-th one is $f(i)$ modulo $998\,244\,353$.

Standard Input	Standard Output

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

3
1 2 3
2 3 1
3 1 2
	6 13 34

5
1 4 2 5 3
2 5 1 3 4
300000000 100000000 500000000 400000000
200000000
	600000000 303511294 612289529 324650937
	947905622

Note

In the second example:

- Consider permutation $[1, 2, 3]$. Indices $1, 2, 3$ are prefix maximums. Index 3 is the only suffix maximum. Indices $2, 3$ are ascents. In conclusion, it has 3 prefix maximums, 1 suffix maximums, and 2 ascents. Therefore, its cost is $a_3b_1c_2 = 12$.
- Permutation $[1, 3, 2]$ has 2 prefix maximums, 2 suffix maximums, and 1 ascent. Its cost is 6.
- Permutation $[2, 1, 3]$ has 2 prefix maximums, 1 suffix maximum, and 1 ascent. Its cost is 4.
- Permutation $[2, 3, 1]$ has 2 prefix maximums, 2 suffix maximums, and 1 ascent. Its cost is 6.
- Permutation $[3, 1, 2]$ has 1 prefix maximum, 2 suffix maximums, and 1 ascent. Its cost is 3.
- Permutation $[3, 2, 1]$ has 1 prefix maximum, 3 suffix maximums, and 0 ascents. Its cost is 3.

The sum of all permutations' costs is 34, so $f(3) = 34$.