ICPC India 2025-26 Online Round

Memory limit is 1.5GB for all problems.

A - How many?

A college club is going to host an ICPC-style contest for its college students. Each team participating in the contest should have **exactly** three participants.
There are n teams participating in the contest. How many participants are there in total?

Input

- The only line of input contains a single integer n, denoting the number of teams that will be participating.

Output

- Output a single integer: the total number of participants.

Constraints

- $1 \leq n \leq 1000$

Sample Input 1

```
1
```

Sample Output 1

```
3
```

Time Limit : 1 second

B - Pseudo Palindrome

You are given an array a of length n and a non-negative integer d.
Is it possible to rearrange a in such a way that $|a_i - a_{n+1-i}| \le d$ for all i $(1 \le i \le n)$?
Here, $|x - y|$ denotes the absolute difference between x and y.
For example, $|3 - 1| = 2, |5 - 9| = 4, |6 - 6| = 0$.

Input

- The first line of input will contain a single integer T, denoting the number of test cases.
- Each test case consists of two lines of input.
  - The first line of each test case contains two space-separated integers n and d — the length of the array and the maximum allowed difference.
  - The second line of each test case contains n space-separated integers $a_1, a_2, \ldots, a_n$ — the elements of the array.

Output

- For each test case, output on a new line the answer: 'Yes' if a valid rearrangement exists, and 'No' otherwise.
- Each character of the output can be printed in either uppercase or lowercase, i.e. the strings 'NO', 'no', 'No', and 'nO' will all be treated as equivalent.

Constraints

- $1 \le T \le 1000$
- $1 \le n \le 2000$
- $1 \le a_i \le 10^9$
- $0 \le d \le 10^9$
- The sum of n over all test cases won't exceed 2000.

Sample Input 1

```
3
1 1
1
2 0
1 2
2 1000
1 2
```

Sample Output 1

```
YES
NO
YES
```

Explanation

- **Test case 1:** The given array is already valid.
- **Test case 2:** It can be proven that there is no valid rearrangement of a.
- **Test case 3:** The valid rearrangements are $[1, 2]$ and $[2, 1]$.

Time Limit : 2 seconds

C - XOR LCM

You are given a positive integer c ($1 \le c \le 10^7$).
You need to find any two positive integers a and b such that:

- $1 \le a, b \le 10^{17}$
- $(a \oplus c) + (b \oplus c) = \mathrm{lcm}(a, c) + \mathrm{lcm}(b, c)$.

Here, $\oplus$ denotes the bitwise XOR operator and $\mathrm{lcm}(x, y)$ denotes the lowest common multiple of x and y.
It can be proven that it is always possible to find a and b under the given constraints. If there are multiple possible solutions, you may find any one of them.

Input

- Each input file contains multiple test cases. The first line of input will contain a single integer T, denoting the number of test cases.
- Each test case consists of a single line, containing one integer c.

Output

- For each test case, output the two integers a and b you found, separated by a space. a and b must satisfy the conditions laid out in the statement.
- If there are multiple possible solutions, any one of them will be accepted.

Constraints

- $1 \le T \le 10^5$
- $1 \le c \le 10^7$

Sample Input 1

```
3
1
2
7
```

Sample Output 1

```
88 71
80 62
1 35
```

Time Limit : 2 seconds

D - Make Empty

You are given a permutation p of $[1, 2, \ldots, n]$. It is guaranteed that n is even.
You can perform the following operation on it:

- Select a subsequence † (say t) of length $2k$ (k does not need to be the same across operations) such that either $\max(t_1, t_2, \ldots, t_k) < \min(t_{k+1}, t_{k+2}, \ldots, t_{2k})$, or $\min(t_1, t_2, \ldots, t_k) > \max(t_{k+1}, t_{k+2}, \ldots, t_{2k})$.
- Then, remove every element of t from p.

You want to make p empty using the **minimum** number of operations.
You must also print the subsequence used in each operation. Note that you must **print the values** (not the indices).

If there are multiple ways to make p empty in the minimum number of operations, you may output any of them.

† A sequence x is a subsequence of a sequence y if x can be obtained from y by deleting several (possibly, zero or all) elements. For example, $[1, 3]$, $[1, 2, 3]$, and $[2, 3]$ are subsequences of $[1, 2, 3]$. On the other hand, $[3, 1]$ and $[2, 1, 3]$ are not subsequences of $[1, 2, 3]$.

Input

- The first line of input will contain a single integer T, denoting the number of test cases.
- Each test case consists of two lines of input.
  - The first line of each test case contains a single integer n — the size of the permutation.
  - The second line of each test case contains n space-separated integers $p_1, p_2, \ldots, p_n$ — the elements of the permutation.

Output

For each test case,

- First, print the minimum number of operations needed, say x, on a single line.
- Then, print x lines denoting the operations you are going to perform.
- On each line, first output the length of the subsequence, followed by the elements of a valid subsequence.
- Note that you must **print the values, not the indices.**

Constraints

- $1 \leq T \leq 10^4$
- $2 \leq n \leq 2 \cdot 10^5$
- n is even.
- p is a permutation of $[1, 2, \ldots, n]$.
- The sum of n over all test cases won't exceed $2 \cdot 10^5$.

Sample Input 1

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

Sample Output 1

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

Explanation

- **Test case** 1**:** We can select $t = p$ and make p empty in one operation.
- **Test case** 2**:** We can again select $t = p$ in the first operation. We can see that t is valid because $\max(t_1, t_2) = 2$ and $\min(t_3, t_4) = 3$, and $2 < 3$.
- **Test case** 3**:** It is not possible to make p empty in one operation because $\max(p_1, p_2) > \min(p_3, p_4)$ and $\min(p_1, p_2) < \max(p_3, p_4)$. In the first operation, we can select $t = [2, 4]$. On removing t from p, we get $p = [3, 1]$. So, we can select $t = p$ in the second operation.

Time Limit : 3 seconds

E - Counting is Fun

You are given a tree with n nodes and a positive integer c. The i-th edge connects the nodes u_i and v_i.
Before the traversal begins, you must choose a permutation p of $[1, 2, \ldots, n-1]$ and label the i-th edge with p_i. These labels remain fixed throughout the traversal.
You start from node 1 with an integer $x = 1$.
During the traversal, you can repeatedly perform one of the following operations:

- **Operation A:** Traverse an edge adjacent to your current node if it is **not** labelled with x. After traversing, you move to the node on the other end of that edge.
- **Operation B:** Increment x by 1.

You must start at node 1, visit all nodes, and **return** to node 1. The **cost** of the traversal is the number of times you perform **Operation B**.
Define $f(p)$ as the minimum cost of a traversal when the edges are labelled according to the permutation p. Your task is to count the number of permutations p of $[1, 2, \ldots n-1]$ such that $f(p) = c$. Since the number might be large, output it modulo $998\,244\,353$.

Input

- The first line of input will contain a single integer T, denoting the number of test cases.
- Each test case consists of multiple lines of input.
  - The first line of each test case contains two space-separated integers n and c — the number of vertices in the tree and the target cost, respectively.
  - The next $n-1$ lines describe the edges. The i-th of these $n-1$ lines contains two space-separated integers u_i and v_i, denoting that the i-th edge is between u_i and v_i.

Output

For each test, find the number of valid permutations modulo $998\,244\,353$.

Constraints

- $1 \le T \le 10^4$
- $2 \le n \le 2 \cdot 10^5$
- $1 \le c \le n-1$
- $1 \le u_i < v_i \le n$
- The input edges describe a tree on n vertices.
- The sum of n over all test cases won't exceed $2 \cdot 10^5$.

Sample Input 1

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

Sample Output 1

```
1
24
0
```

Explanation

- **Test case 1:** There is only one permutation, and it is valid.
- **Test case 2:** In the second test case, all $(n-1)!$ permutations are valid.
- **Test case 3:** In the third test case, it can be proven that no permutation is valid.

Time Limit : 3 seconds

F - Non Unique

You are given an array a of length n. Let $f(a, i)$ denote the number of indices j such that $1 \le j < i$ and $a_j > a_i$. Let $F(a)$ be the set of indices i having the maximum value of $f(a, i)$.
You do not want $F(a)$ to consist of only one element, so you may perform the following operation any number of times:

- Select an index i ($1 \le i < n$), and swap a_i and a_{i+1}.

Let x be the minimum number of operations needed so that $|F(a)| \ge 2$. It can be proven that x is finite.
Find the number of ways to achieve $|F(a)| \ge 2$ in exactly x operations.
Since the number might be large, output it modulo $998\,244\,353$.
Two ways are different if there exists an index j such that the indices selected in the j-th operation are different.

Input

- The first line of input will contain a single integer T, denoting the number of test cases.
- Each test case consists of two lines of input.
  - The first line of each test case contains a single integer n — the length of the array.
  - The second line of each test case contains n space-separated integers $a_1, a_2, \ldots, a_n$ — the elements of the array.

Output

For each test case, print the number of ways modulo $998\,244\,353$.

Constraints

- $1 \le T \le 10^5$
- $2 \le n \le 10^6$
- $1 \le a_i \le n$
- The sum of n over all test cases won't exceed 10^6.

Sample Input 1

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

Sample Output 1

```
1
2
2
```

Explanation

- **Test case 1:** We do not need to perform any operation. So, the number of ways is 1.
- **Test case 2:** We can get $|F(a)| \geq 2$ by performing 1 operation. There are 2 ways: we can select $i = 1$ or $i = 2$.
- **Test case 3:** We can achieve $|F(a)| \geq 2$ by performing 2 operations. There are 2 ways:
  - Select $i = 1$ in the first operation and $i = 2$ in the second operation. Then we get $a = [3, 1, 2]$ and $F(a) = [2, 3]$.
  - Select $i = 2$ in the first operation and $i = 1$ in the second operation. Then we get $a = [1, 2, 3]$ and $F(a) = [1, 2, 3]$.

Time Limit : 4 seconds