A. Verify Password

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

Monocarp is working on his new site, and the current challenge is to make the users pick strong passwords.

Monocarp decided that strong passwords should satisfy the following conditions:

- password should consist only of lowercase Latin letters and digits;
- there should be no digit that comes after a letter (so, after each letter, there is either another letter or the string ends);
- all digits should be sorted in the non-decreasing order;
- all letters should be sorted in the non-decreasing order.

Note that it's allowed for the password to have only letters or only digits.

Monocarp managed to implement the first condition, but he struggles with the remaining ones. Can you help him to verify the passwords?

Input

The first line contains a single integer t ($1 \le t \le 1000$) — the number of testcases.

The first line of each testcase contains a single integer n ($1 \le n \le 20$) — the length of the password.

The second line contains a string, consisting of exactly n characters. Each character is either a lowercase Latin letter or a digit.

Output

For each testcase, print "YES" if the given password is strong and "NO" otherwise.

Standard Input	Standard Output
5 4 12ac 5 123wa 9 allllmost 5 ac123 6 011679	YES NO YES NO YES

Note

In the second testcase, the letters are not sorted in the non-decreasing order.

In the fourth testcase, there is a digit that comes after a letter — digit '1' after a letter 'c'.

,

B. Increase/Decrease/Copy

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

You are given two integer arrays: array a of length n and array b of length $n + 1$.

You can perform the following operations any number of times in any order:

- choose any element of the array a and increase it by 1;
- choose any element of the array a and decrease it by 1;
- choose any element of the array a, copy it and append the copy to the end of the array a.

Your task is to calculate the minimum number of aforementioned operations (possibly zero) required to transform the array a into the array b. It can be shown that under the constraints of the problem, it is always possible.

Input

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

Each test case consists of three lines:

- the first line contains a single integer n ($1 \le n \le 2 \cdot 10^5$);
- the second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$);
- the third line contains $n + 1$ integers $b_1, b_2, \ldots, b_{n+1}$ ($1 \le b_i \le 10^9$).

Additional constraint on the input: the sum of n over all test cases doesn't exceed $2 \cdot 10^5$.

Output

For each test case, print a single integer — the minimum number of operations (possibly zero) required to transform the array a into the array b.

Standard Input	Standard Output
3 1 2 1 3 2 3 3 3 3 3 4 4 2 1 2 2 1 5 2 3	3 1 8

Note

In the first example, you can transform a into b as follows: $[2] \to [2, 2] \to [1, 2] \to [1, 3]$.

,

C. Job Interview

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

Monocarp is opening his own IT company. He wants to hire n programmers and m testers.

There are $n + m + 1$ candidates, numbered from 1 to $n + m + 1$ in chronological order of their arriving time. The i-th candidate has programming skill a_i and testing skill b_i (a person's programming skill is different from their testing skill). The skill of the team is the sum of the programming skills of all candidates hired as programmers, and the sum of the testing skills of all candidates hired as testers.

When a candidate arrives to interview, Monocarp tries to assign them to the most suitable position for them (if their programming skill is higher, then he hires them as a programmer, otherwise as a tester). If all slots for that position are filled, Monocarp assigns them to the other position.

Your task is, for each candidate, calculate the skill of the team if everyone except them comes to interview. Note that it means that exactly $n + m$ candidates will arrive, so all $n + m$ positions in the company will be filled.

Input

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

Each test case consists of three lines:

- the first line contains two integers n and m ($0 \le n, m \le 2 \cdot 10^5$; $2 \le n + m + 1 \le 2 \cdot 10^5$) — the number of programmers and the number of testers Monocarp wants to hire, respectively;
- the second line contains $n + m + 1$ integers $a_1, a_2, \ldots, a_{n+m+1}$ ($1 \le a_i \le 10^9$), where a_i is the programming skill of the i-th candidate;
- the third line contains $n + m + 1$ integers $b_1, b_2, \ldots, b_{n+m+1}$ ($1 \le b_i \le 10^9$; $b_i \neq a_i$), where b_i is the testing skill of the i-th candidate.

Additional constraint on the input: the sum of $(n + m + 1)$ over all test cases doesn't exceed $2 \cdot 10^5$.

Output

For each test case, print $n + m + 1$ integers, where the i-th integer should be equal to the skill of the team if everyone except the i-th candidate comes to interview.

Standard Input	Standard Output
4 1 0 2 1 1 2 0 2 4 5 5 5 4 1 1 2 2 1 5 4 5 2 3 1	1 2 5 6 9 8 11 11 12 13 13 13 12 15

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

Note

Let's consider the third test case of the example:

- if the 1-st candidate does not arrive, the 2-nd candidate gets hired as a tester, the 3-rd candidate gets hired as a programmer, the 4-th candidate gets hired as a tester. The total skill of the team will be $2 + 5 + 1 = 8$;
- if the 2-nd candidate does not arrive, the 1-st candidate gets hired as a tester, the 3-rd candidate gets hired as a programmer, the 4-th candidate gets hired as a tester. The total skill of the team will be $5 + 5 + 1 = 11$;
- if the 3-rd candidate does not arrive, the 1-st candidate gets hired as a tester, the 2-nd candidate gets hired as a tester, the 4-th candidate gets hired as a programmer. The total skill of the team will be $5 + 2 + 4 = 11$;
- if the 4-th candidate does not arrive, the 1-st candidate gets hired as a tester, the 2-nd candidate gets hired as a tester, the 3-rd candidate gets hired as a programmer. The total skill of the team will be $5 + 2 + 5 = 12$.

,

D. Invertible Bracket Sequences

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

A regular bracket sequence is a bracket sequence that can be transformed into a correct arithmetic expression by inserting characters '1' and '+' between the original characters of the sequence. For example:

- bracket sequences "()()" and "(())" are regular (the resulting expressions are: "(1)+(1)" and "((1+1)+1)");
- bracket sequences ")(", "(" and ")" are not.

Let's define the *inverse* of the bracket sequence as follows: replace all brackets '(' with ')', and vice versa (all brackets ')' with '('). For example, strings "()((" and ")())" are inverses of each other.

You are given a regular bracket sequence s. Calculate the number of pairs of integers (l, r) ($1 \le l \le r \le |s|$) such that if you replace the substring of s from the l-th character to the r-th character (inclusive) with its inverse, s will still be a regular bracket sequence.

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 test case contains a non-empty regular bracket sequence; it consists only of characters '(' and/or ')'.

Additional constraint on the input: the total length of the regular bracket sequences over all test cases doesn't exceed $2 \cdot 10^5$.

Output

For each test case, print a single integer — the number of pairs (l, r) meeting the conditions from the statement.

Standard Input	Standard Output
4 (()) () ()()() (()())(())	1 0 3 13

Note

In the first example, there is only one pair:

- $(2, 3)$: (()) $\rightarrow$ ()().

In the second example, there are no pairs.

In the third example, there are three pairs:

- $(2, 3)$: ()()() $\rightarrow$ (())();
- $(4, 5)$: ()()() $\rightarrow$ ()(());

- $(2, 5)$: ( ) ( ) ( ) $\rightarrow$ (( ) ( ));

,

E. Splittable Permutations

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

Initially, we had one array, which was a permutation of size n (an array of size n where each integer from 1 to n appears exactly once).

We performed q operations. During the i-th operation, we did the following:

- choose any array we have with at least 2 elements;
- split it into two non-empty arrays (prefix and suffix);
- write two integers l_i and r_i, where l_i is the maximum element in the left part which we get after the split, and r_i is the maximum element in the right part;
- remove the array we've chosen from the pool of arrays we can use, and add the two resulting parts into the pool.

For example, suppose the initial array was $[6, 3, 4, 1, 2, 5]$, and we performed the following operations:

1. choose the array $[6, 3, 4, 1, 2, 5]$ and split it into $[6, 3]$ and $[4, 1, 2, 5]$. Then we write $l_1 = 6$ and $r_1 = 5$, and the arrays we have are $[6, 3]$ and $[4, 1, 2, 5]$;
2. choose the array $[4, 1, 2, 5]$ and split it into $[4, 1, 2]$ and $[5]$. Then we write $l_2 = 4$ and $r_2 = 5$, and the arrays we have are $[6, 3]$, $[4, 1, 2]$ and $[5]$;
3. choose the array $[4, 1, 2]$ and split it into $[4]$ and $[1, 2]$. Then we write $l_3 = 4$ and $r_3 = 2$, and the arrays we have are $[6, 3]$, $[4]$, $[1, 2]$ and $[5]$.

You are given two integers n and q, and two sequences $[l_1, l_2, \ldots, l_q]$ and $[r_1, r_2, \ldots, r_q]$. A permutation of size n is called *valid* if we can perform q operations and produce the given sequences $[l_1, l_2, \ldots, l_q]$ and $[r_1, r_2, \ldots, r_q]$.

Calculate the number of valid permutations.

Input

The first line contains two integers n and q ($1 \le q < n \le 3 \cdot 10^5$).

The second line contains q integers $l_1, l_2, \ldots, l_q$ ($1 \le l_i \le n$).

The third line contains q integers $r_1, r_2, \ldots, r_q$ ($1 \le r_i \le n$).

Additional constraint on the input: there exists at least one permutation which can produce the given sequences $[l_1, l_2, \ldots, l_q]$ and $[r_1, r_2, \ldots, r_q]$.

Output

Print one integer — the number of valid permutations, taken modulo 998244353.

Standard Input	Standard Output
6 3 6 4 4 5 5 2	30

10 1 10 9	1814400
4 1 2 4	8

,

F. Remove Bridges

```
Input file:     standard input
Output file:    standard output
Time limit:     2 seconds
Memory limit:   256 megabytes
```

You are given a rooted tree, consisting of n vertices, numbered from 1 to n. Vertex 1 is the root. Additionally, the root only has one child.

You are asked to add exactly k edges to the tree (possibly, multiple edges and/or edges already existing in the tree).

Recall that a bridge is such an edge that, after you remove it, the number of connected components in the graph increases. So, initially, all edges of the tree are bridges.

After k edges are added, some original edges of the tree are still bridges and some are not anymore. You want to satisfy two conditions:

- for every bridge, all tree edges in the subtree of the lower vertex of that bridge should also be bridges;
- the number of bridges is as small as possible.

Solve the task for all values of k from 1 to $n - 1$ and output the smallest number of bridges.

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 a single integer n ($2 \le n \le 3 \cdot 10^5$) — the number of vertices of the tree.

Each of the next $n - 1$ lines contain two integers v and u ($1 \le v, u \le n$) — the description of the edges of the tree. It's guaranteed that the given edges form a valid tree.

Additional constraint on the input: the root (vertex 1) has exactly one child.

The sum of n over all testcases doesn't exceed $3 \cdot 10^5$.

Output

For each testcase, print $n - 1$ integers. For each k from 1 to $n - 1$ print the smallest number of bridges that can be left after you add k edges to the tree.

Standard Input	Standard Output
4 2 1 2 12 4 10 5 12 12 11 3 6 9 6 1 6	0 7 3 1 0 0 0 0 0 0 0 0 4 1 0 0 0 0 0 0 0 0 0

12 7 11 6 2 11 10 9 10 8 8 1 2 2 3 2 4 3 5 3 6 4 7 4 8 5 1 2 2 3 3 4 4 5	