The 2019 ICPC Asia Shanghai Regional Contest Online Contest

Hosted by School of Computer Engineering and Science, Shanghai University

15^{th} September 2019

Problem A. Lightning Routing I

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	512 megabytes

The Lightning Network is a "Layer 2" payment protocol that operates on top of a blockchain-based cryptocurrency (like Bitcoin). It enables fast transactions between participating nodes and has been touted as a solution to the Bitcoin scalability problem. It features a peer-to-peer system for making micropayments of cryptocurrency through a network of bidirectional payment channels without delegating custody of funds.

In this problem, we'll regard the Lightning Network as a tree. And due to the network environment, the cost of these payment channels are changeable. That says, you are given a weighted undirected tree on n vertices and q queries. Each changes the weight of one edge or query the furthest distance of one vertex to another vertex. The distance between two vertices is the sum of the weights on the unique simple path that connects them.

Input

The first line contains an integer n ($1 \leq n \leq 10^5$) — the number of vertices in the tree.

Following $n-1$ lines will contain the initial structure of the tree. The i-th of these lines contains three space-separated integers x_i, y_i, w_i ($1 \leq x_i, y_i \leq n, 0 \leq w_i \leq 10^9$) meaning that initially, there is an edge between vertices x_i and y_i with weight w_i. It is guaranteed that these $n-1$ lines describe a tree.

The next line contains an integer q ($1 \leq q \leq 10^5$) — the number of qeries. Finally, q lines describing queries follow. The i-th of these lines contains either 'Q v_i' or 'C e_i w_i'.

- C e_i w_i ($1 \leq e_i \leq n-1, 0 \leq w_i \leq 10^9$), changes the weight of the e_i-th edge to w_i, or
- Q v_i ($1 \leq v_i \leq n$), query the furthest distance of the vertex v_i to another vertex.

Output

Output the query for each query.

Sample input and output

Sample Input	Sample Output
5 1 2 1 1 3 1 2 4 1 2 5 1 3 Q 1 C 1 2 Q 2	2 3

Problem B. Light bulbs

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	8 megabytes

There are N light bulbs indexed from 0 to $N-1$. Initially, all of them are off.

A FLIP operation switches the state of a contiguous subset of bulbs. $FLIP(L, R)$ means to flip all bulbs x such that $L \leq x \leq R$. So for example, $FLIP(3, 5)$ means to flip bulbs 3 , 4 and 5, and $FLIP(5, 5)$ means to flip bulb 5.

Given the value of N and a sequence of M flips, count the number of light bulbs that will be on at the end state.

Input

The first line of the input gives the number of test cases, T. T test cases follow. Each test case starts with a line containing two integers N and M, the number of light bulbs and the number of operations, respectively. Then, there are M more lines, the i-th of which contains the two integers L_i and R_i, indicating that the i-th operation would like to flip all the bulbs from L_i to R_i, inclusive.

$1 \leq T \leq 1000$

$1 \leq N \leq 10^6$

$1 \leq M \leq 1000$

$0 \leq L_i \leq R_i \leq N-1$

Output

For each test case, output one line containing Case #x: y, where x is the test case number (starting from 1) and y is the number of light bulbs that will be on at the end state, as described above.

Sample input and output

Sample Input	Sample Output
2 10 2 2 6 4 8 6 3 1 1 2 3 3 4	Case #1: 4 Case #2: 3

Problem C. Triple

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	128 megabytes

You are given three arrays A, B, and C, each with N integers. Please count the number of triplets (i, j, k) (with $1 \leq i, j, k \leq N$) that can satisfy the following three conditions:

- $|A_i - B_j| \leq C_k$, and
- $|B_j - C_k| \leq A_i$, and
- $|A_i - C_k| \leq B_j$

Input

The first line of the input gives the number of test cases, T ($T \leq 100$).

Each test case begins with an integer N ($1 \leq N \leq 10^5$): the number of integers in arrays A, B, and C.

The second line consists of N integers A_i ($1 \leq A_i \leq 10^5$).

The third line consists of N integers B_i ($1 \leq B_i \leq 10^5$).

The fourth line consists of N integers C_i ($1 \leq C_i \leq 10^5$).

There are at most 20 test cases with $N > 1000$.

Output

For each test case, output one line containing Case #x: y, where x is the test case number (starting from 1) and y is the number of triplets satisfying the conditions given in the problem statement.

Sample input and output

Sample Input	Sample Output
2 3 1 2 3 1 2 3 1 2 3 4 1 1 2 4 1 1 3 5 1 2 5 7	Case #1: 24 Case #2: 31

Problem D. Counting Sequences I

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 seconds
Memory limit:	8 megabytes

We define a sequence $a_1, a_2, ..., a_n$ of positive integer numbers, as perfect sequence if it matches following qualities:

- $n \geq 2$, and
- $a_1 + a_2 + ... + a_n = a_1 \times a_2 \times ... \times a_n$

Input

The first line contains the number of test cases T (no more than 300). In each of the following T lines there is one integer $n(2 \leq n \leq 3000)$.

Output

For each test case, print the number of perfect sequences that has length n. In case the result is too large, print the remainder modulo 1000000007.

Sample input and output

Sample Input	Sample Output
2 2 3	1 6

Note

n = 2: {2, 2}

n = 3: {1, 2, 3}, {1, 3, 2}, {2, 1, 3}, {2, 3, 1}, {3, 1, 2}, {3, 2, 1}

Problem E. Counting Sequences II

Input file:	standard input
Output file:	standard output
Time limit:	1 second
Memory limit:	128 megabytes

You are given two integers n and m, find out how many sequences $a_1, a_2, ..., a_n$ of non-negative integer numbers satisfy the following signatures:

- $1 \leq a_i \leq m$ for $1 \leq i \leq n$, and
- $count(i)\%2 = 0$ for all $i\%2 = 0$, $count(i)$ is the number of integers in the sequence that equals i.

Input

The first line contains the number of test cases T (no more than 30). In each of the following T lines there are two integers $n(1 \leq n \leq 10^{18})$ and $m(1 \leq m \leq 200000)$.

Output

For each test case, print the number of sequences. In case the result is too large, print the remainder modulo 1000000007.

Sample input and output

Sample Input	Sample Output
2 3 2 2 3	4 5

Note

For $n = 3$, $m = 2$, all sequences are: {1, 1, 1}, {1, 2, 2}, {2, 1, 2}, {2, 2, 1}

For $n = 2$, $m = 3$, all sequences are: {1, 1}, {2, 2}, {3, 3}, {1, 3}, {3, 1}

Problem F. Rhyme scheme

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	64 megabytes

A rhyme scheme is the pattern of rhymes at the end of each line of a poem or song. It is usually referred to by using letters to indicate which lines rhyme; lines designated with the same letter all rhyme with each other.

e.g., the following "poem" of 4 lines has an associated rhyme scheme "ABBA"

1 — 99 bugs in the code A

2 — Fix one line B

3 — Should be fine B

4 — 100 bugs in the code A

This essentially means that line 1 and 4 rhyme together and line 2 and 3 rhyme together.

The number of different possible rhyme schemes for an n-line poem is given by the Bell numbers. For example, $B_3 = 5$, it means there are five rhyme schemes for a three-line poem: AAA, AAB, ABA, ABB, and ABC.

The question is to output the k-th rhyme scheme in alphabetical order for a poem of n lines. For example: the first rhyme scheme of a three-line poem is "AAA", the fourth rhyme scheme of a three-line poem is "ABB".

Input

The first line of the input gives the number of test cases, T ($1 \leq T \leq 10000$). T test cases follow.

Each test case contains a line with two integers n and k.

$1 \leq n \leq 26, 1 \leq k \leq B_n$ (B_n is the n-th of Bell numbers)

Output

For each test case, output one line containing Case #x: y, where x is the test case number (starting from 1) and y is the rhyme scheme contains uppercase letters.

Sample input and output

Sample Input	Sample Output
7 1 1 2 1 3 1 3 2 3 3 3 4 3 5	Case #1: A Case #2: AA Case #3: AAA Case #4: AAB Case #5: ABA Case #6: ABB Case #7: ABC

Problem G. Substring

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	20 megabytes

Word A can match word B if their corresponding starting and ending letters are same and the sets of letters in between are either the same or permutations of each other. For example "study" can match "sdtuy", "sduty", "stduy". But "study" can not match "stuy", "tsudy".

Given a string S containing lowercase letters, you need to answer M queries, each query contains a word W. Your task is to count the number of nonempty substrings of S that matches word W.

Input

The first line of the input gives the number of test cases, T ($T \leq 20$).

The first line contains a string S only contains lowercase. The length of the string will not be greater than $100,000$.

The second line contains one integer M ($1 \leq M \leq 20000$): the number of queries.

The following M lines, each line contains a word W only contains lowercase. The length of word W won't more than $100,000$, and won't less than 2. The total length of strings in M queries has the limit of $100,000$.

Output

For each query, the output should contain a single number, on a single line: the number of nonempty substrings of S that matches word W.

Sample input and output

Sample Input	Sample Output
1 abccdefgdaaacdcdfegaada 4 gadaa abccd defg aa	2 1 2 3

Problem H. Luhhy's Matrix

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	128 megabytes

Luhhy recently learned matrix multiplication. Formally, if A is an $n \times m$ matrix and B is an $m \times p$ matrix, their matrix product C is an $n \times p$ matrix such that $C_{i,j} = \sum_{k=0}^{m-1} A_{i,k} B_{k,j}$ for $i = 0, \ldots, n-1$ and $j = 0, \ldots, p-1$. Now she is trying to solve a problem about that. Luhhy hates big numbers, so she only calculates the result of matrix product modulo 2 in this problem.

She has a queue which is empty at first. She has to perform N operations.

Each operation is either PUSH or POP.

- For PUSH operation, she is given a 16×16 matrix A and then puts it to the tail of the queue.
- For POP operation, she takes out the head from the queue.

After PUSH or POP operation, she has to compute products of all the remaining matrices in queue. Luhhy knows that the order of two matrices matters in matrix multiplication and for some reason she chooses to compute products from the tail to head. Luhhy is puzzled by this hard problem and now she asks you to help.

To avoid massive input dataset, matrix A is attained by the following algorithm:

Input a random *seed* (unsigned int):

```
seed ^= lastans;
for(int i = 0; i < 16; ++i){
    seed ^= seed * seed + 15;
    for(int j = 0; j < 16; ++j){
        A[i][j] = (seed >> j) & 1;
    }
}
```

where *lastans* (unsigned int) denotes the answer after the last operation and is initially zero.

To avoid massive output, you should output:

$$\sum_{i=0}^{15} \sum_{j=0}^{15} M_{i,j} \times 17^i \times 19^j \bmod 2^{32}$$

where M denotes the products of all the remaining matrices.

You may assume that when executing POP, the queue isn't empty. In particular, if the queue is empty after POP operation, you should just output 0.

Input

The input contains several test cases, and the first line contains a single integer T ($1 \le T \le 5$), the number of test cases.

For each test case, the first line contains an integer N ($1 \leq N \leq 50\,000$), indicating the number of operations.

Each of the following N lines contains two integers t and $seed$ ($1 \leq t \leq 2, 0 \leq seed < 2^{32}$), $t = 1$ for PUSH operation and $t = 2$ for POP operation (in this case $seed = 0$).

Output

For each query, output an integer in one line.

Sample input and output

Sample Input	Sample Output
1 5 1 123456789 1 0 2 0 1 123456789 2 0	3512312066 1360316010 3870620630 2686897404 2195405248

Problem I. Debug

Input file:	`standard input`
Output file:	`standard output`
Time limit:	10 seconds
Memory limit:	8 megabytes

Kblack wrote a program, which consisted of n lines of code, executing sequentially. Each line produced a result which would be used in the next line. However, there was a bug in exactly one line of his code, so that every result since this line was wrong. In order to find this bug, he decided to add `print` after several lines and run the entire program several times, alternatively.

Formerly, if Kblack added a `print` after line k, he could know whether the bug was before line k (include k) or after line k. The problem here, was, he didn't know how to do this efficiently. Namely, he needed a seconds to run the entire code and b seconds to add a `print`. It's obviously time-wasting to add `print` after every line, or add only one `print` each time and run it many times.

It's left up to you to figure out how much time he needs at least in the worst case.

Input

The first line an integer $T(1 \leq T \leq 10^4)$, which is the number of cases.

In the next T lines, each line contains three integers n, a, b $(1 \leq n, a, b \leq 10^{18})$, which means the time of each run and the time of adding a print line.

Output

For all T lines, each line is one integer, denoting the least time needed in the worst case.

Sample input and output

Sample Input	Sample Output
2 5 1 2 5 2 1	8 6

Note

In the first example, Kblack ran the code twice. For the first time he `print` after line 3. In the worst case, the bug was before line 3, so he added `print` line 1 and line 2 for the second run and then he could locate the bug.

In the second example, Kblack `print` the line 1, 2, 3, 4 and only needed to run once to fix the bug.

Problem J. Stone game

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

CSL loves stone games. He has n stones; each has a weight a_i. CSL wants to get some stones. The rule is that the pile he gets should have a higher or equal total weight than the rest; however if he removes any stone in the pile he gets, the total weight of the pile he gets will be no higher than the rest. It's so easy for CSL, because CSL is a talented stone-gamer, who can win almost every stone game! So he wants to know the number of possible plans. The answer may be large, so you should tell CSL the answer modulo $10^9 + 7$.

Formerly, you are given a labelled multiset $S = \{a_1, a_2, \ldots, a_n\}$, find the number of subsets of S: $S' = \{a_{i_1}, a_{i_2}, \ldots, a_{i_k}\}$, such that

$$(Sum(S') \geq Sum(S - S')) \wedge (\forall t \in S', Sum(S') \leq Sum(S - S') - t)\,.$$

Input

The first line an integer T ($1 \leq T \leq 10$), which is the number of cases.

For each test case, the first line is an integer n ($1 \leq n \leq 300$), which means the number of stones. The second line are n space-separated integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 500$).

Output

For each case, a line of only one integer t — the number of possible plans. If the answer is too large, please output the answer modulo $10^9 + 7$.

Sample input and output

Sample Input	Sample Output
2 3 1 2 2 3 1 2 4	2 1

Note

In example 1, CSL can choose the stone 1 and 2 or stone 1 and 3.

In example 2, CSL can choose the stone 3.

Problem K. Peekaboo

Input file:	`standard input`
Output file:	`standard output`
Time limit:	3 seconds
Memory limit:	256 megabytes

Tension is in the air! You are playing a game Peekaboo with Kblack and CSL. All of you are in a 2D grid. You are at point $(0, 0)$. Kblack and CSL has hidden themselves at some lattice point (x-coordinate and y-coordinate are both integers) out of your sight, but you have the psycho connection that tells you the distance between you and Kblack is a, the distance between you and CSL is b and the distance between Kblack and CSL is c. You now want to know all the possible positions of Kblack and CSL.

Input

The first line is an integer T ($1 \le T \le 20$), which is the number of cases.

The next T lines each have three space-separated integers a, b, c ($1 \le a, b, c \le 10^9$), respectively denoting the distance in the problem statement.

Output

For each case, you should output firstly in a line, an integer n, which is the number of possible positions.

For the next n lines, each line contains four space-separated integers x_1, y_1, x_2, y_2 which are the Kblack's position and CSL's position. In particular, the possible positions should follow the **lexicographical order**. The data guarantees that $\sum n \le 10^5$.

Sample input and output

Sample Input	Sample Output
2 1 1 1 3 4 5	0 8 -3 0 0 -4 -3 0 0 4 0 -3 -4 0 0 -3 4 0 0 3 -4 0 0 3 4 0 3 0 0 -4 3 0 0 4

Problem L. Digit sum

Input file:	`standard input`
Output file:	`standard output`
Time limit:	2 seconds
Memory limit:	128 megabytes

A digit sum $S_b(n)$ is a sum of the base-b digits of n. Such as $S_{10}(233) = 2 + 3 + 3 = 8$, $S_2(8) = 1 + 0 + 0 = 1$, $S_2(7) = 1 + 1 + 1 = 3$.

Given N and b, you need to calculate $\sum_{n=1}^{N} S_b(n)$.

Input

The first line of the input gives the number of test cases, T. T test cases follow. Each test case starts with a line containing two integers N and b.

$1 \leq T \leq 10000$

$1 \leq N \leq 10^6$

$2 \leq b \leq 10$

Output

For each test case, output one line containing Case #x: y, where x is the test case number (starting from 1) and y is answer.

Sample input and output

Sample Input	Sample Output
2 10 10 8 2	Case #1: 46 Case #2: 13