Gellyfish and Priority Queue

Input file:	`standard input`
Output file:	`standard output`
Time limit:	5 seconds
Memory limit:	1024 megabytes

There is a min-priority queue Q, initially empty. Gellyfish will perform m operations on it. Each belongs to one of the two types:

- **Insertion**: Given l, r. Gellyfish will generate a uniformly random integer x from the range $[l, r]$ and then insert it into Q.
- **Extraction**: Delete the current minimum element in Q. It's guaranteed that Q is not empty.

After all the operations are done, Gellyfish wants you to foretell the expected product of all the remaining elements in Q, taking modulo 998244353.

Input

Each test contains multiple test cases. The first line contains the number of test cases t ($1 \le t \le 200$). The description of the test cases follows.

The first line of each test case contains two integers m and v ($1 \le m \le 500$, $1 \le v \le 500$).

Then m lines follow; the i-th contains one or three integers, denoting an operation. If the operation is an insertion, then there will be three integers $o = 1$, l, r ($1 \le l \le r \le v$), separated by space; if the operation is an extraction, then the input line consists of a single integer $o = 2$.

It is guaranteed that the sum of m over all test cases does not exceed 2000. It's also guaranteed that the priority queue is not empty when performing an extraction operation.

Output

For each test case, output a single line containing an integer: the expected product of all the remaining elements in the priority queue, modulo 998244353.

Example

standard input	standard output
5 2 1 1 1 1 2 3 4 1 1 4 1 1 4 2 5 6 1 1 3 1 4 6 2 1 2 4 2 7 8 1 1 6 1 2 4 2 1 1 5 1 3 7 2 1 2 8 9 10 1 3 8 1 2 9 1 1 6 2 1 2 5 2 1 7 10 1 4 8 2	1 873463812 5 255107002 624423011

Note

In the first example test case, note that the product of all the elements in an empty priority queue is considered as 1.

In the second example test case, two random integers, each from the range $[1, 4]$, are put in the priority queue after the first two operations. Then, the smaller one is deleted, making the product of all the remaining elements equal to the larger integer. Therefore, the expected product is $\frac{1}{16} \cdot (1+2+3+4+2+2+3+4+3+3+3+4+4+4+4+4) = \frac{25}{8}$, and $\frac{25}{8} \equiv 873463812 \pmod{998244353}$.

In the third example test case, since $3 < 4$ and $4 \leq 4$, the second inserted integer is always the maximum element in the priority queue. Since eventually the size of the priority queue becomes 1, it's sufficient to see the expected answer is the expectation of the second inserted number, which is $\frac{1}{3} \cdot (4 + 5 + 6) = 5$.