Problem M: Median

Time limit: 20s, memory limit: 1GB.

Yaawn... you really shouldn't have played on ByteStation till 3 AM. You begin to slowly drift away, immersed by the monotone voice of your algorithms professor...

Wait. What happened? It seems that you fell asleep during online classes, and slept through the entire afternoon, night, and the next morning. But that means... the exam! Barely catching your breath, you bust into the classroom precisely when the professor is beginning to explain the problem statement.

The task at the exam is to compute the median of a sequence, defined as the middle element in the sorted order (if the length of the sequence is even, the median is the smaller out of the two middle values). You're supposed to write down the pseudocode of a solution, and then simulate it on an example sequence provided by the professor.

Reaching into the depths of your memory, you seem to recall something like that appearing during the lecture. Some sort of a magic algorithm... magic threes? Yaawn, the memory itself makes you sleepy again. You somehow split the sequence into parts, solve them recursively, and then combine...?

Based on the bits and pieces that you remembered, you came up with the following algorithm:

```
function magicThrees(sequence)
  if the length of the sequence is no more than 2 then
    return the smallest value in sequence
  otherwise
    part_1, part_2, part_3 = splitIntoThreeParts(sequence)
    median_i = magicThrees(part_i) dla i = 1, 2, 3
    return the median of [median_1, median_2, median_3]
```

where `splitIntoThreeParts` divides the sequence into three connected fragments, with lengths as close to each other as possible. Concretely, the fragments will have lengths `[s, s, s]`, `[s + 1, s, s]` or `[s + 1, s + 1, s]`, depending on the length of the original sequence. For example, the sequence `[8, 2, 6, 6, 3, 5, 7, 1]` will be divided into `[8, 2, 6]`, `[6, 3, 5]` and `[7, 1]`.

After leaving the exam, you realized that your algorithm is not so magical after all, as it doesn't always work. *Maybe, at least, it has worked on the example sequence from the exam...* Unfortunately, your memory of that sequence is as fuzzy as the one of the algorithm itself: while you do remember *almost all* elements of the sequence from the exam, you're not sure about a few of the values. However, you do remember the overall bounds on all values appearing in the sequence: all elements were supposed to lie in a (closed) interval $[0, m-1]$.

Calculate the number of ways to choose the values you don't know, in such a way that `magicThrees` executed on the resulting sequence returns the correct median (as defined above). As the answer may be very large, it's enough if you find its remainder modulo $10^9 + 7$.

Input

The first line of input contains the number of test cases z. The descriptions of the test cases follow.

The first line of a test case contains two integers n, m ($n \geqslant 1$, $1 \leqslant m \leqslant 10^9$), denoting the length of the test sequence and the bound on the values of its elements.

The second line contains the test sequence, described as n integers from the range $[-1, m-1]$, where elements with unknown values are denoted by -1.

Tests

If we denote the number of unknown values by q, then every test case belongs to one of the following groups:

- $1 \leqslant z \leqslant 100$, $1 \leqslant q \leqslant 10$, $n \leqslant 3^4 = 81$
- $z = 15$, $1 \leqslant q \leqslant 20$, $n \leqslant 3^5 = 243$
- $z = 3$, $q = 30$, $n \leqslant 3^8 = 6\,561$

Output

For every test case, output a single integer r ($0 \leqslant r < 10^9 + 7$) – the answer to the question posed in the problem statement.

Example

For an example input:	the correct output is:
3 3 10 -1 -1 3 4 50 10 20 -1 40 5 100 -1 10 10 -1 20	100 21 1979

Explanation

In the first test case, `magicThrees` returns the correct median irrespective of the two unknown values; the answer is thus $10^2 = 100$.

In the second test case, `magicThrees` returns the correct median if and only if the unknown value is not larger that 20, which gives 21 ways.

In the third test case, `magicThrees` returns an *incorrect* median if both unknown values are smaller than 10, or both larger, which gives $100^2 - (10^2 + 89^2) = 1979$ ways.