2 Stack Permutations

There are a number of sorting algorithms such as heap sort etc. that are used in the world today. But there aren't many unsorting algorithms. Pradator wonders why this is so, and proposes to build his own unsorting algorithm.

Pradator is also fed up with using complicated data structures such as heaps; he likes to write nice clean code and heaps make coding dirty. So he decides instead to use just one stack for his unsorting algorithm. Here is how his unsorting algorithm works.

Initially, he has the numbers $1, 2, \ldots, N$ in that order in a list and the stack is empty. In each step he can either remove the top most element of the stack and print it out on the output (this is permitted only if the stack is nonempty) or remove the next element from the list and push it on top of the stack. He continues performing such operations till both the stack and the list are empty (and thus some permutation of $1, \ldots, N$ is printed on the output).

Pradator thinks that any permutation of $1, \ldots, N$ can be printed out by choosing an appropriate sequence of operations. However, his friend Rudy quickly finds a counterexample: when $N = 3$ the permutation $3, 1, 2$ cannot be produced in this manner. Since 3 has to appear first, you have no choice but to push all three elements one after the other into the stack before removing 3 and printing it on the output and this rules out $1, 2$ as a possibility for the rest of the output. Let us call a permutation to be a *stack permutation* if it can be produced using Pradator's algorithm. You can check that the permutations $1, 2, 3$ and $3, 2, 1$ can be produced using Pradator's algorithm. Rudy claims that given any permutation, he will be able to tell you if it is a stack permutation or not and moreover he can also tell how many such stack permutations are lexicographically less than a given stack permutation. (A permutation $x_1, x_2, \ldots, x_N$ is lexicographically smaller than $y_1, y_2, \ldots, y_N$ if for the smallest j such that $x_j \neq y_j$ we have $x_j < y_j$.)

Pradator now feels left behind and needs your help to help him do all that his friend Rudy claims to be able to do. You should write a program that takes a list of queries, where each query is a permutation of $1, \ldots, N$, and answers them as follows: The answer to a query is -1 if the given permutation is not a stack permutation. Otherwise, you should report the number of stack permutations strictly less than the current permutation modulo $(10^9 + 7)$ (since the answer may be too large).

Input format

- The first line contains two integers, giving the number N and the number of queries Q.
- This is followed by Q lines, giving Q queries. The ith query is described in line $i + 1$ and contains a permutation of $1, \ldots, N$.

Output format

Q lines of output where line i contains a single integer giving the answer to the ith query (remember answers for valid stack permutations must be reported modulo $(10^9 + 7)$).

Test Data

- Subtask 1 (10 marks) : $1 \leq N \leq 10$, $Q = 1$.
- Subtask 2 (15 marks) : $1 \leq N \leq 13$, $Q = 1$.
- Subtask 3 (35 marks) : $1 \leq N \leq 5000$, $Q = 1$.
- Subtask 4 (40 marks) : $1 \leq N \leq 5000$, $Q \leq 2000$.

Sample Input

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

Sample Output

```
-1
4
```

Limits

- *Time limit:* 5 seconds
- *Memory limit:* 128 MB