Sorting - 100 points

Instead of partying in the Bellagio, Johnnie spends his time sorting permutations. He chooses a random permutation having *N* elements. While this permutation is not sorted, Johnnie picks two consecutive elements and swaps them. After a while he learns how to sort any permutation using the algorithm above with a minimum number of swaps.

Last night Johnnie learnt a new algorithm, so now he can swap **any** two elements in the permutation. Johnnie will sort the permutation using a minimum number of swaps. Obviously, this new algorithm will use less or the same number of swaps as the old one.

You have to find how many permutations having *N* elements can be sorted with fewer swaps using the new algorithm. You should print the result modulo *999017*.

Input Data
On the first line of the standard input there is one number *N*, the length of permutations.

Output Data
Print the answer modulo *999017* on the first line of the standard output.

Restrictions
2 <= N <= 1000

Example

Standard Input	Standard Output	Explanation
4	**11**	There are 11 permutations with the mentioned property: 1 4 3 2 2 4 3 1 3 2 1 4 3 2 4 1 3 4 1 2 3 4 2 1 4 1 3 2 4 2 1 3 4 2 3 1 4 3 1 2 4 3 2 1 For example, the permutation 4 2 1 3 can be sorted using the first algorithm like this: **4 2** 1 3 => 2 **4 1** 3 => **2 1** 4 3 => 1 2 **4 3** => 1 2 3 4. Four steps were necessary. Using the second algorithm, it can be sorted faster: 4 2 **1 3** => **4** 2 3 **1** => 1 2 3 4. Only two steps were necessary.

Time limit: 1 second / test case