Twin Permutations - Hackerearth

Sunday, February 23, 2020 9:23 AM

Also known as "Ambiguous Permutations"

The Problem:

Define a permutation of length *N* as a sequence of *N* numbers, consisting of all the numbers from *1* to *N* in any order.
An inverse permutation of a permutation (let say P) is a sequence of numbers in which the i^{th} number is the position of number i in the original permutation (*permutation P)* , $1 \le i \le N$.

For example -
For a permutation 2 5 1 4 3, inverse permutation is 3 1 5 4 2 .
Given a number *N*, find the number of *distinct permutations* of length *N* which can not be distinguished from their inverse permutation.
PS : Large I/O.

Input:
First line consists of a single integer *T*, the number of test cases.
Next *T* lines consist of a single integer each, representing value of *N* for that test case.

Constraints:
$1 \le T \le 10^6$
$1 \le N \le 10^6$

Output:
For each testcase, output the desired answer *modulo* 10^9+7 in a separate line.

Sample Input:	Sample Output:
1 3	4

Explanation
There are 6 permutations of length 3. They are written along with their inverse permutation below.

Permutation	Inverse Permutation
1 2 3	1 2 3
1 3 2	1 3 2
2 1 3	2 1 3
2 3 1	3 1 2
3 1 2	2 3 1
3 2 1	3 2 1

Clearly, there are 4 such permutations that can not be distinguished from their inverse permutation.

The Code:

```
#include <stdio.h>
#define MOD 1000000007
#define M 1000000


int main(){
	int caseCount, N;
	long int permutations[M];
	scanf("%d", &caseCount);

	permutations[0] = 1;
	permutations[1] = 2;
	for(int i=2; i<=M; i++){
	   permutations[i] = ( permutations[i-1] + permutations[i-2] * 1LL * (i) ) % MOD;
	}

	while(caseCount > 0){
	   scanf("%d", &N);
	   printf("%ld\n", permutations[N-1]);
	   caseCount--;
	}
}
```

The Stats:

Score
30.0

Time (sec)
0.30824

Memory (KiB)
8008

Language
C