C - Factorial

Input: `factorial.in`
Output: `standard output`

The most important part of a GSM network is so called *Base Transceiver Station (BTS)*. These transceivers form the areas called *cells* (this term gave the name to the cellular phone) and every phone connects to the BTS with the strongest signal (in a little simplified view). Of course, BTSes need some attention and technicians need to check their function periodically.

ACM technicians faced a very interesting problem recently. Given a set of BTSes to visit, they needed to find the shortest path to visit all of the given points and return back to the central company building. Programmers have spent several months studying this problem but with no results. They were unable to find the solution fast enough. After a long time, one of the programmers found this problem in a conference article. Unfortunately, he found that the problem is so called "Travelling Salesman Problem" and it is very hard to solve. If we have n BTSes to be visited, we can visit them in any order, giving us $n!$ possibilities to examine. The function expressing that number is called factorial and can be computed as a product $1 \times 2 \times 3 \times 4 \times \cdots \times n$. The number is very high even for a relatively small n.

The programmers understood they had no chance to solve the problem. But because they have already received the research grant from the government, they needed to continue with their studies and produce at least some results. So they started to study behaviour of the factorial function.

For example, they defined the function z. For any positive integer n, $z(n)$ is the number of zeros at the end of the decimal form of number $n!$. They noticed that this function never decreases. If we have two numbers $n_1 < n_2$, then $z(n_1) \leq z(n_2)$. It is because we can never "lose" any trailing zero by multiplying by any positive number. We can only get new and new zeros. The function z is very interesting, so we need a computer program that can determine its value efficiently.

Input

There is a single positive integer t on the first line of input. It stands for the number of numbers to follow. Then there is t lines, each containing exactly one positive integer number n, $1 \leq n \leq 1000000000$.

Output

For every number n, output a single line containing the single non–negative integer $z(n)$.

Sample Input

```
6
3
60
100
1024
```

23456
8735373

Sample Output

0
14
24
253
5861
2183837