

# Totient in permutation (easy)

In number theory, Euler's totient (or PHI function), is an arithmetic function that counts the number of positive integers less than or equal to a positive integer  $N$  that are relatively prime to this number  $N$ .

That is, if  $N$  is a positive integer, then  $\text{PHI}(N)$  is the number of integers  $K$  for which  $\text{GCD}(N, K) = 1$  and  $1 \leq K \leq N$ . We denote  $\text{GCD}$  the Greatest Common Divisor. For example, we have  $\text{PHI}(9)=6$ .

Interestingly,  $\text{PHI}(87109)=79180$ , and it can be seen that 87109 is a permutation of 79180.

## Input

The input begins with the number  $T$  of test cases in a single line.

In each of the next  $T$  lines there are an integer  $M$ .

## Output

For each given  $M$ , you have to print on a single line the value of  $N$ , for which  $1 < N < M$ ,  $\text{PHI}(N)$  is a permutation of  $N$  and the ratio  $N/\text{PHI}(N)$  produces a minimum. If there's several answers output the greatest, or if need, "No solution." without quotes.

Leading zeros are not allowed for integers greater than 0.

## Example

### Input:

```
3
22
222
2222
```

### Output:

```
21
63
291
```

**Explanations :** For the first case, in the range  $]1..22[$ , the lonely number  $n$  for which  $\text{phi}(n)$  is in permutations( $n$ ) is 21, (we have  $\text{phi}(21)=12$ ). So the answer is obviously 21.

For the second case, in the range  $]1..222[$ , there's two numbers  $n$  for which  $\text{phi}(n)$  is in permutations( $n$ ), we have  $\text{phi}(21)=12$  and  $\text{phi}(63)=36$ . But as  $63/36$  is equal to  $21/12$ , we're taking the greater : 63.

For the third case, in the range  $]1..2222[$ , there's four numbers  $n$  for which  $\text{phi}(n)$  is in permutations( $n$ ),  $\text{phi}(21)=12$ ,  $\text{phi}(63)=36$ ,  $\text{phi}(291)=192$  and  $\text{phi}(502)=250$ . Within those solutions  $291/192$  is the minimum, we output 291.

## Constraints

$1 < T < 10^5$   
 $1 < M < 10^7$

Code size limit is 10kB ; less than 500B of python3 code can get AC under 2s.

After that you may try [TIP2](#).

@Speed addicts : my C code ran in 0.02s, and my fastest python3.2 code ran in 1.21s, (0.90s in py2.7)

Edit 2017-02-11, after compiler updates. My old C code ends in 0.00s, my old Python code ends in 0.05s !!!