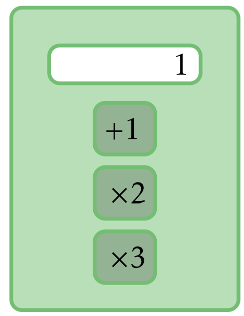
5.2.2 Primitive Calculator

Primitive Calculator Problem
Find the minimum number of operations needed to get a positive integer n from 1 by using only three operations: add 1, multiply by 2, and multiply by 3.

> **Input:** An integer n.
> **Output:** The minimum number of operations "+1", "×2", and "×3" needed to get n from 1.



You are given a calculator that only performs the following three operations with an integer x: add 1 to x, multiply x by 2, or multiply x by 3. Given a positive integer n, your goal is to find the minimum number of operations needed to obtain n starting from the number 1. Before solving the programming challenge below, test your intuition with our Primitive Calculator puzzle.

Let's try a greedy strategy for solving this problem: if the current number is at most $n/3$, multiply it by 3; if it is larger than $n/3$, but at most $n/2$, multiply it by 2; otherwise add 1 to it. This results in the following pseudocode.

```
GreedyCalculator(n):
numOperations ← 0
currentNumber ← 1
while currentNumber < n:
   if currentNumber ≤ n/3:
      currentNumber ← 3 × currentNumber
   else if currentNumber ≤ n/2:
      currentNumber ← 2 × currentNumber
   else:
      currentNumber ← 1 + currentNumber
   numOperations ← numOperations + 1
return numOperations
```

Stop and Think. Can you find a number n such that

$$\textsc{GreedyCalculator}(n)$$

produces an incorrect result?

Input format. An integer n.

Output format. In the first line, output the minimum number k of operations needed to get n from 1. In the second line, output a sequence of intermediate numbers. That is, the second line should contain positive integers $a_0, a_1, \ldots, a_k$ such that $a_0 = 1$, $a_k = n$ and for all $1 \le i \le k$, a_i is equal to either $a_{i-1} + 1$, $2a_{i-1}$, or $3a_{i-1}$. If there are many such sequences, output any one of them.

Constraints. $1 \le n \le 10^6$.

Sample 1.

Input:

```
1
```

Output:

```
0
1
```

Sample 2.

Input:

```
96234
```

Output:

```
14
1 3 9 10 11 22 66 198 594 1782 5346 16038 16039 32078 96234
```

Another valid output in this case is "1 3 9 10 11 33 99 297 891 2673 8019 16038 16039 48117 96234".