Problem A. Zeros

We are interested in computing the number of zeros at the end of the number $\operatorname{lcm}(a, a+1, ..., b)$, that is, of the least common multiple of numbers $a, a+1, ..., b$.

Subtask 1

When $b \le 16$, we can explicitly compute the lcm that Santa is interested in. The maximum value that it can attain is $\operatorname{lcm}(1, 2, ..., 16) = 16 \cdot 9 \cdot 5 \cdot 7 \cdot 11 \cdot 13 = 720\,720$.

One way is to use the facts that $\operatorname{lcm}(x, y, z) = \operatorname{lcm}(\operatorname{lcm}(x, y), z)$ and that $\operatorname{lcm}(x, y) = \frac{x \cdot y}{\gcd(x,y)}$, where $\gcd(x, y)$ is the greatest common divisor of x and y and can be computed using a built-in library function (e.g. `__gcd` in C++) or using the Euclidean algorithm.

Another possibility is to compute the lcm directly from the definition – loop over every positive integer and check if it is a multiple of all of $a, a+1, ..., b$. (Return the first number that passes this test.)

Once we have computed the lcm, we can find its number of trailing zeros by dividing it by 10 as many times as possible.

Subtask 2

Here we can still compute the lcm explicitly. However, we must use the first method above, as now the maximum value is $\operatorname{lcm}(1, 2, ..., 40) = 5\,342\,931\,457\,063\,200 \approx 5 \cdot 10^{15}$ and we cannot afford to make this many iterations. Moreover, we must take care to use a 64-bit integer variable type (such as `long long` in C++).

Subtask 3

One solution is to still compute the lcm explicitly – but now the result can have up to 90 digits, so we need to either implement our own big-number type, or use Python, which has that built in. However, there is a much more elegant solution, which consists only of several `if` statements.

We can see that the answer is 0 for $b = 1$, it can only increase as b grows, and it only increases at powers of 5, by 1 at a time. Thus we can answer 0 for $b \in \{1, 2, 3, 4\}$, 1 for $b \in \{5, ..., 24\}$, 2 for $b \in \{25, ..., 124\}$, and 3 for $b \in \{125, ..., 200\}$. But why is that? To see this, let us make some simple observations (which will be useful also for the following subtasks).

For two positive integers q and x, let us denote by $v_q(x)$ the multiplicity of q as a divisor of x, that is, the maximum $k \in \mathbb{Z}_{\ge 0}$ such that q^k divides x. Then we have:

- $v_{10}(x)$ is the number of zeros at the end of x.
- $v_{10}(x) = \min(v_2(x), v_5(x))$. This is because $10^k \mid x$ if and only if $2^k \mid x$ and $5^k \mid x$.
- For a prime number p, $v_p(\operatorname{lcm}(x, y)) = \max(v_p(x), v_p(y))$. This is because if we decompose both x and y into prime factors: $x = p_1^{k_1} \cdot ... \cdot p_s^{k_s}$ and $y = p_1^{\ell_1} \cdot ... \cdot p_s^{\ell_s}$, where $p_1, ..., p_s$ are different primes and $k_1, ..., k_s, \ell_1, ..., \ell_s \in \mathbb{Z}_{\ge 0}$, then $\operatorname{lcm}(x, y) = p_1^{\max(k_1,\ell_1)} \cdot ... \cdot p_s^{\max(k_s,\ell_s)}$.
- As a consequence,

$$v_{10}(\operatorname{lcm}(a, a+1, ..., b)) = \min\left[\max(v_2(a), v_2(a+1), ..., v_2(b)), \max(v_5(a), v_5(a+1), ..., v_5(b))\right].$$

For this subtask, the solution follows by noting that when $a = 1$, then $\max(v_2(1), v_2(2), ..., v_2(b))$ is the largest k such that $2^k \le b$ (in other words, it is $\lfloor \log_2 b \rfloor$). The term corresponding to 5 is the largest k such that $5^k \le b$, and of course this second k is no larger, as $2^k \le 5^k$ (equivalently, $\log_5 b \le \log_2 b$). This value of k is 0 for $b \in \{1, ..., 5-1\}$, 1 for $b \in \{5, ..., 5^2-1\}$, 2 for $b \in \{5^2, ..., 5^3-1\}$, and 3 for $b \in \{5^3, ..., 200\}$ (since $200 < 5^4$).

Subtask 4

Now the input numbers are becoming large (remember to read them as 64-bit integers). However, the condition $b - a \le 10^6$ means that we can use the $\min[\max(...), \max(...)]$ formula above explicitly, by looping over all $i = a, ..., b$. For each such i we compute $v_2(i)$ directly, by dividing i by 2 as many times as possible; this will take at most $\log_2(10^{18})$ steps; and similarly for 5.

Subtask 5

We simply generalize the solution of subtask 3 above: instead of several `if` statements, we compute $k = \lfloor \log_5 b \rfloor$ by starting from $k = 0$ and increasing it as long as $5^k \leq b$.

Subtask 6

To solve the general task, we will compute the expression $\max(v_2(a), v_2(a+1), ..., v_2(b))$ (and similarly for 5) faster than in $O(b - a)$ time. That is, we want to find the largest k such that 2^k divides some number $i \in \{a, ..., b\}$. Clearly, this reduces to checking whether a given k is good, as there are only at most $\log_2(10^{18})$ different k-values to check. To find out whether the interval $\{a, ..., b\}$ contains some multiple of 2^k, we can, for example, round b down to the nearest multiple of 2^k, and check if a is still no larger than the rounded b. More formally, we compute the largest multiple of 2^k that is at most b (this is $\lfloor \frac{b}{2^k} \rfloor \cdot 2^k$) and check whether it is no smaller than a. If yes, then it is a multiple of 2^k that lies in the interval $\{a, ..., b\}$; otherwise, there is no such multiple.