Documentation

The problem involves determining whether a given integer is an ugly number. An ugly number is a positive integer whose prime factors are limited to 2, 3, and 5. This problem can be solved by repeatedly dividing the number by 2, 3, and 5 until no longer divisible by them. If the result is 1, the number is considered ugly, otherwise, it is not. The solution must handle a range of integers, including negative numbers, zero, and positive numbers, and provide an efficient method to check if a number is ugly.

The first step in the solution is to check for invalid inputs. Since ugly numbers must be positive, any number less than or equal to zero is immediately deemed not ugly. This prevents unnecessary calculations and ensures that the solution handles edge cases effectively, such as zero or negative integers, which do not qualify as ugly numbers by definition.

Next, the solution proceeds by checking if the given number can be divided by 2, 3, and 5. This is done by iterating over these primes and continuously dividing the number by each of them as long as it is divisible. This step reduces the number by eliminating its prime factors 2, 3, and 5, which is key to determining if the number only consists of these prime factors. By dividing the number in this manner, we can efficiently factor out these primes and simplify the problem.

After performing the division operations, the final step involves checking whether the number has been reduced to 1. If the number is reduced to 1, it means that the only prime factors of the original number were 2, 3, or 5, making it an ugly number. If the result is greater than 1, then the number must have contained other prime factors, and it is not considered an ugly number. This final check ensures that the solution correctly identifies ugly numbers and distinguishes them from non-ugly numbers.

The time complexity of this approach is logarithmic with respect to the value of the number, as each division by 2, 3, or 5 reduces the number significantly. This makes the solution efficient and scalable, even for larger inputs. By handling negative numbers, zero, and positive integers systematically, the solution provides a clear and concise way to determine if a number is ugly, ensuring both accuracy and efficiency in solving the problem.