Assignment: *Powerize* (Compulsory)

Apply Test-Driven Development to implement the function `powerize(int)` with specification

```
/**
 * Writes a number as a power with maximal exponent.
 *
 * @param n  the number to 'powerize'
 * @throws IllegalArgumentException  if precondition violated
 * @pre 2 <= n
 * @post n == \result.base ^ \result.exponent &&
 *     (\forall int b, int e;
 *      2 <= b && 1 <= e && n == b ^ e;
 *      e <= \result.exponent)
 */
public static Power powerize(int n)
        throws IllegalArgumentException
```

where `Power` is a *record* class briefly defined by

```
public static class Power {
    public int base; // 0 <= base
    public int exponent; // 0 <= exponent
}
```

Use the given skeleton code.

- Of course, you also apply *functional decomposition*, and supply contracts and test cases for all auxiliary functions.
- Your auxiliary functions do *not* have to be robust.
- Do pay attention to performance (run time, memory). The call `powerize(n)` should return its result within 10 seconds.
- Do *not* use floating point numbers.

Submit your program, test cases, and archive in Momotor.

Hint Your solution must be based on factorizing n. For instance, use *trial division* with candidate divisors d such that $2 \leq d$ and $d^2 \leq n$, to write $n = p_1^{e_1} \times p_2^{e_2} \times \cdots \times p_k^{e_k}$.

Note that each prime power can be stored in a `Power` object. Thus, the factorization can be stored in a `List<Power>` object.

Next, find the *greatest common divisor* g of all exponents e_i. Then, we have $n = \left(p_1^{e_1/g} \times p_2^{e_2/g} \times \cdots \times p_k^{e_k/g}\right)^g$. Therefore, g is the maximal exponent and the corresponding base is $p_1^{e_1/g} \times p_2^{e_2/g} \times \cdots \times p_k^{e_k/g}$.

Submit your work to Momotor.

In each submitted file containing your work, write:

- full author name,
- id number,
- date of latest change.

In a program text you, obviously, do this in a comment.

In text files (including program code), write author information above the *cut line*, being the first line containing the *cut mark* `--8<--`.