**This problem will help you remember the idea of implementing Comparable in java** (readSW 1.4)

**Problem 1.** *(Die.java)* Implement a data type Die in Die.java that represents a six-sided die and supports the following API:



$ java Die 3 3 5

\* \*

\*

\* \*

false

true

**These problems will help you understand the Analysis of Algorithms** (read CLRS)

**Problem 2.** (True or False): No comparison-based sorting algorithm can do better than *Ω(n log n)* in the worst-case

**Problem 3.** We can extend our notion to the case of two parameters n and m that can go to infinity independently at different rates. For a given function *g(n, m)*, we donate by *O(g(n, m))* the set of functions

*O(g(n,m))* = { *f(n, m)* : there exist positive constants *c, n0*, and *m0*,

such that *0 <= f(n, m) <= c g(n, m)* for all *n >= n0* or *m >= m0*}

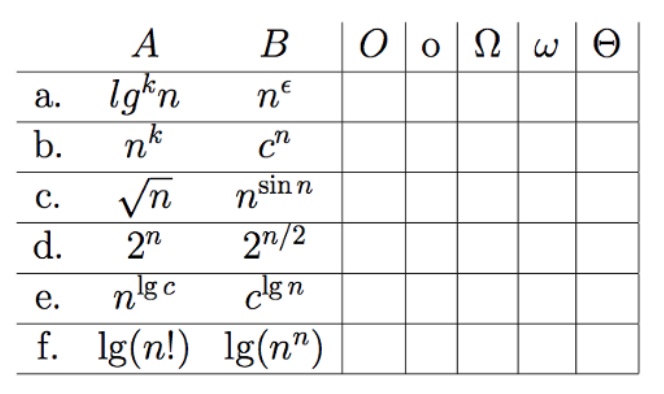
Give corresponding definitions for *Ω(g(n, m))* and *Θ(g(n, m))*

**Problem 4.** Consider functions *f(n)* and *g(n)* as given below. Use the most precise asymptotic notation to show how function *f* is related to function g in each case ( i.e. , *f ∈ ?(g)*). For example, if you were given the pair of functions *f (n) = n* and *g(n) = n2* then the correct answer would be: *f ∈ o(g)*. To avoid any ambiguity between *O(g)* and *o(g)* notations due to writing, use *Big-O(g)* instead of *O(g).*

A screenshot of a cell phone

Description automatically generated

**Problem 5.** Indicate, for each pair of expressions (A, B) in the table below, whether A is O, Ω, Θ, o, ω of B. Assume that k>= 1, ϵ > 0 and *c* >1 are constant. Your answer should be in the form of the table with “yes” or “no” written in each box.



**Submitting Information:**

* Use the code I provided for each problem. DON’T DELETE ANY FUNCTION
* You should have each problem in a separate .java files (ex: Die.java, hw2.pdf).
* Submit your work on Canvas in one Zip file.
* The deadline is Monday, Sep 23st at 5:59PM