CPSC 3200 Object-Oriented Development

Programming Assignment #2: Due Thursday April 14, 2022 before MIDNIGHT

P2 exercises your understanding of composition, ownership and C++ copy semantics

For an acceptable P2 submission:

1. **use C++:** the g++ compiler (C++17) on cs1
   a. Submissions that do not compile on cs1 will NOT receive credit
   b. programs developed in Visual Studio often do NOT compile on cs1
   c. HIGHLY recommended to use C++ tools: g++, CLion, Xcode etc.
2. **upload all .h and .cpp files to Canvas (no zip files AND do not use .hpp)**
3. **use submission script to upload and compile on cs1**
   /home/fac/dingle/submit/22sq3200/p2_runme
4. design using heap memory, **composition,** and **move semantics**
   a. `gridFlea` reused from P1 but rewritten in C++
   b. the extent to which your P1 design is used as is or modified is your choice
      but consider design comments given in 3200 class sessions
   c. with the variable cardinality of `gridFlea` subobjects across the `infest` objects,
      use heap memory: do NOT design using excessive capacity.
5. Fulfill requirements as specified in steps 4-7 from P1

Part I: Class Design

Each *inFest* object encapsulates some number of internally generated, distinct *gridFlea* objects; the cardinality of *gridFlea* subobjects thus varies across *inFest* objects. For example, constructor `inFest(x,y,z)` could use the first two parameters to instantiate its first *gridFlea* subobject and then systematically generate different values for its remaining `z-1` *gridFlea* subobjects. Each *inFest* object manages its *gridFlea* subobjects, supporting the ability to move all *gridFlea* objects, e.g. `j.move(x)` triggers `g.move(x)` for all *gridFlea* objects encapsulated in *inFest* object `j`. Each *inFest* object also supports client queries as to the minimum value acquired from a `value()` call across all *gridFlea* subObjects, the maximum value acquired from a `value()` call across all *gridFlea* subObjects. An *inFest* object must respond consistently (restore? change state?) when more than half of its *gridFlea* subobjects are inactive or deactivated.

Deep copying must be supported. Move semantics must be defined. <u>Do not use a rng inside *inFest*</u>

Many details are missing. ***You MUST make and DOCUMENT your design decisions***
Do NOT tie your type definition to the Console.

Part II: Driver (P2.cpp) -- External Perspective of Client – tests your class design

Fulfill the testing requirements as specified in P1 -- a collection of `inFest` objects
Unit testing is not required or expected.
Demonstrate copying of `inFest` objects via call by value and assignment