**CPSC 3200 Object-Oriented Development** Programming Assignment #4

P4 design requires operator overloading*and modern* C++ *memory management*

1. Augment type definitions, as noted below
2. Expand the interfaces of MsgStream, DurableStream, PartitionStreamto include the overloading of appropriate operators.
   1. **Focus on external (client) expectations for use of type**
   2. Remember type definition is concerned with consistency.

Use Contractual design for documentation.  **DO NOT hard code**

**CPSC 3200 Programming Assignment 4**

**Part I: Class Design**

**DurableStream in C++**

**Re-write the DurableStream class in C++. You do not need to write dispose(), as ofstream’s destructor automatically closes the file. You also do not need to add your own buffer for file output as ofstream automatically buffers output. Update your definition of MsgStream to support inheritance.**

**Smart Pointers and Rule of 5 Methods**

**Replace all raw pointers in your classes with smart pointers. Choose the most appropriate type of smart pointer for the given situation.**

**With the introduction of smart pointers, you may no longer have the need for manual memory management. Remove your destructors methods as appropriate.**

**Suppress copying of the PartitionStream class. You may choose whether to allow or suppress copying of DurableStream, but you must justify your decision in your implementation invariants. Copying of MsgStream carries over from P2, with any needed modifications.**

**You may not use STL containers (vector, map, etc.) but you may use the STL smart pointers.**

**Implement at least six operators in your classes.**

**You must:**

**Add operators to at least two classes.**

**Create at least two destructive and two non-destructive operators. The remaining (at least two) operators can be either destructive or non-destructive.**

**Create at least one unary operator and one binary operator. The other (at least four) operators can be either unary or binary.**

**Create operators from at least of the three following categories (the other, at least three, operator can be selected from any category):**

**Arithmetic operators (+, -, \*, /, %, +=, -=, \*=, /=, %=).**

**(In)equality operators (== and !=). If you implement one then you must implement the other. (== and !=) count as only one operator.**

**Relational operators (<, >). Note that combined operators must also be implemented when appropriate, e.g. if you implement == and < then you also need to implement <= (which still counts as only two overloaded operators).**

**The indexing operator ([]).**

**Stream insertion or extraction (<<, >>).**

**Consider existing methods when writing overloaded operators; some methods may already perform the behavior expected of an operator. Overloading an expected operator in an appropriate manner increases abstraction and readability.**

**Add a justification for each operator to your implementation invariants, explaining why that operator is appropriate for the class.**

**Part II: Driver --** unit testing is not required or expected.

Do NOT use cin in your driver.

Driver specifics:

1. Design a ***functionally decomposed*** driver to demonstrate program requirements
2. Illustrate **(test):**
   1. Design modifications
   2. Overloaded operators – invoke either directly in the driver or indirectly
   3. Correct implementation of deep copying
3. Smart pointers must be used
4. **Instantiate several instances of PartitionStream with a variety of different configurations of MsgStreams (all MsgStreams, a mix of MsgStreams and DurableStreams, just one stream, etc.).**
5. **Test all of the operators you implemented.**

Upload your individual files to Canvas: NO zip files