Multi-threaded element queue project (C++/Cmake)

1. Develop a class from scratch to queue a finite number of primitive types (e.g., int). This class will be used for multi-threaded communication as follows:

- Reading thread pops elements while Writing thread pushes elements.
- Two popping methods shall be available. If queue is empty, one of them shall block indefinitely, while the other one should throw an exception after a given timeout if no element is pushed meanwhile.
- If queue is full, pushing a new element shall drop the oldest element in-queue before storing a new one.
- To support different types, a template class shall be implemented.
- Dynamic memory allocation shall be used to store queue elements (no std library for storing elements).
- Race conditions shall be avoided.
- Code shall be document. Preferably using Doxygen style.
- Performance issues are optionally addressed.
- The class interface should look like this:

```
class Queue<T> {
         Queue(int size) {...}
         void Push(T element) {...}
         T Pop() {...}
         T PopWithTimeout(int milliseconds) {...}
         int Count() {...} // Amount of elements stored now
         int Size() {...} // Max number of elements
}
```

As an example implement the following in *main()*:

Writing Thread step	Queue (after step)	Reading Thread step
New Queue<int>(2)		
Push(1)	1	
		Pop() -> 1
Push(2)	2	
Push(3)	2,3	
Push(4) // Element 2 dropped!	3,4	
	4	Pop() -> 3
		Pop() -> 4
		Pop() // blocks
Push(5)	5	
		-> 5 // is released

2. Develop unit tests for the Queue class with support of a framework (e.g. cppunit, gtest, catch).

3. Develop a CMake configuration file to ease the build process (inc. tests).

4. Share your code at your Gitthub or Bitbucket account and send us the link.