Write a class StackAccess that has a stack object of size 6 as its property. The class allows only one thread to read (pop) from and write (push) to the stack at one time. If the stack is empty, a thread wishes to read from the stack will wait until it is not empty. If the stack is full, a thread wishes to write to the stack will wait until it is not full. [Hint: use lock condition and its methods to accomplish this]

Then, write a driver class with a thread pool of 3. The driver will run 5 threads – 3 threads to read from the stack (in an instance of the StackAccess class) every 50 ms for 5 times and 2 threads to write to the stack an arbitrary number between 0 and 99 every 20ms for 10 times.

After completing the above task, use ReadWriteLock to implement locking for a peek method that retrieves the first element from StackAccess without removing it. If the stack is empty, the reading thread will wait until it is not empty and return the value on top of the stack.

NB: The behaviour of StackAccess is different from a normal stack which will return NULL (or an equivalent value) when the stack is empty, or abort the push when the stack is full.