1. It we defined the normal iterator operations as const we would not allow the iterator to be used to modify the elements that it is pointing to. For example, if we created an iterator called “it” over a data type containing integers and tried to perform the operation *it = 5 we would get an error.

2.
    - The copy constructor requires an = operator of the template type.
    - The = operator override requires an = operator of the template type.
    - The pushBack function requires an = operator of the template type.
    - The filter function requires a () operator of the Condition template type. It also requires an iterator for the queue template type and begin and end functions that return an iterator from a queue. The iterator must have ++, == and * operators.
    - Similarly the transform requires a () operator of the Transformation template type. It also requires an iterator for the queue template type and begin and end functions that return an iterator from a queue. The iterator must have ++, == and * operators.

3. The student will receive a compiler error, likely a linker error, that says that it can’t find an instance of the Queue class of a specific type or that it can’t find a template definition when they try to compile the exercise, which indicates that the template definition for the Queue class was not found. This error will occur during the preprocessing stage of compilation.

4. The student can create a function that returns a boolean and receives one parameters called numerator which is of type integer. Outside the function somewhere else in the .cpp, the student can define a global variable called divisor which is of type integer which can be defined during runtime. Inside the function, the student can check if the numerator is divisible by the divisor (the function has access to the numerator because it was passed in as an argument and it also has access to the divisor because the divisor is global) by checking if numerator mod(divisor) is equal to 0 and then return true if so and otherwise false.