Q.A:

Defining the normal iterator operations as const would not be sufficient to allow iteration on a fixed queue, because a const iterator is more restrictive than a normal iterator that has its operations defined as const.

A normal iterator that has its operations defined as const is still capable of modifying the elements in the container, even though the operations themselves are const. This is because the const qualifier only applies to the operations and not to the iterator itself.

Q.B:

- **The pushBack function** assumes that T has a copy constructor, so that it can create a copy of the item parameter and insert it into the queue.
- **The const front function** assumes that T has a default constructor, so that it can return a default-constructed object if the queue is empty.
- **The non-const front function** also assumes that T has an assignment operator, so that it can assign the value of the front item in the queue to the return value.
- **The popFront** function assumes that T has a destructor, so that it can destroy the front item in the queue when it is removed.
- **The begin and end functions** assume that T has a copy constructor, so that they can create copies of the items in the queue when they are returned by the iterators.
- **The *Iterator* class** assumes that T has a default constructor and an assignment operator, so that it can create and assign values to the items it points to.
- **The *ConstIterator* class** assumes that T has a default constructor, so that it can return a default-constructed object if the iterator is dereferenced when it points to the end of the queue.
- **The *Node* constructor** assumes that T has a copy constructor, so that it can create a copy of the item parameter and store it in the m_item member variable.
- **The setNext function** assumes that T has a default constructor, so that it can create a default-constructed object if next is nullptr or this is nullptr.
- **The getRefItem function** assumes that T has an assignment operator, so that it can assign the value of m_item to the return value.

Q.C:

The student will receive the error "undefined reference to `Queue<T>::pushBack()'" (for example) this occurs in the linking stage (a linking error).

This happens because the compiler does not generate any code for the member functions of the class. Instead, it generates a template for the member functions that can be instantiated when the class is used. This means that the implementation of the member functions must be included in the source file where the class is defined, so that the compiler can generate the necessary code.

Q.D:

We can use Functor, this is an example of how to do it:

```cpp
class DivisibleBy
{
public:
    explicit DivisibleBy(int divisor) : m_divisor(divisor) {}

    bool operator()(int x) const { return x % m_divisor == 0; }

private:
    int m_divisor;
};

// ...

Queue<int> q;
for (int i = 1; i < 10; i++)
{
    q.pushBack(i);
}

int divisor = 3;

/**

* here we choose what number we want to use as a divisor (we can use another ways

* to get it like “std::cin” for example

*/

Queue<int> result = filter(q, DivisibleBy(divisor));
```