# CMPUT 350 Lab 6 Prep Problems

1. In file `pp1.cpp` implement class template `Queue` that supports the following standard first-in-first-out queue functions:

```
empty() const   : returns true iff queue is empty
front()         : returns reference to front element (precond.: !empty())
front() const   : returns const reference to front element (precond.: !empty())
pop()           : removes front element (pre-cond.: !empty())
push(x)         : add element x of type T at the end
```

Use private inheritance from `std::list<T>` and delegate function calls to base class function calls, like so:

Examples:

```
template ...
class ...
    : private std::list<T>
{
    using Base = std::list<T>;
    ...
    bool empty() const {
        return Base::empty();
    }
    ...
}
```

Also make sure that your CC, AO, and destructor work (either by convincing yourself that the default implementation works or by writing test code), and check preconditions with assert. Also, test your code.

2. You have decided that for your nuclear power plant control software using floating point variables is a bad idea because of potentially devestating rounding errors. Your idea is to prevent users from instantiating your class templates with floating point types with the help of type trait class `is_fp` and `static_assert(cond, msg)` which checks a Boolean condition at **COMPILE TIME** and prints an error msg if it is false.

Example:

```
template <typename T>
class Foo {
    // don't allow T to be floating point type
    static_assert(...);
    ...
};
cout << is_fp<double>::value << endl;   // 1
cout << is_fp<float>::value << endl;    // 1
cout << is_fp<int>::value << endl;      // 0
Foo<int> f;         // ok
Foo<double> g;      // fails
```

In file `pp2.cpp` implement class template `is_fp<T>` whose variable value is 1 if `T` is a floating point type, and 0 otherwise. Test it using the code above.