# CSCE 622: Generic Programming -- Assignment 1

**Peihong Guo UIN: 421003404**

## Problem 1

There are several types the triple class cannot handle:

1. Incomplete types, such as `void`.
2. Types without default constructors.
3. Types without copy constructors.
4. Types without assignment constructor.
5. Types without equality and inequality operators.

## Problem 2

Adding such a function is a bad idea. The triple template is still usable with arbitrary types after adding this function as long as the following at least one of the following condictions is satisfied:

1. The types used as template parameters must be constructible from integers and support equality operation, so that `is_zero` can be written as:

   ```
   template <typename T1, typename T2, typename T3>
   bool triple<T1, T2, T3>::is_zero() const {
       return first == T1(0) && second == T2(0) && third == T3(0
   );
   }
   ```

2. The types used as template parameters must be comparable against integers, so that `is_zero` can be written as:

```
template <typename T1, typename T2, typename T3>
bool triple<T1, T2, T3>::is_zero() const {
    return first == 0 && second == 0 && third == 0;
}
```

See `p2_test.cpp` for verification.

### Problem 3

```
template <class T>
inline T sum_all(T* first, T* last) {
    T sum;
    for (sum = 0; first != last; ++first)
        sum += *first;
    return sum;
}
```

This function can be instantiated with `int`, but not `std::string`, `void` or a pointer type. To instantiate this function, type `T` must support the following operations: 1. Default construction: `T sum`. 2. Construction from integers: `sum = 0`. 2. Comparision: `first != last` 4. Addition-assignment: `sum += *first`

Consider the 4 types `int`, `std::string`, `void` and a pointer type:

1. `int`: all required operations are well defined for `int`, so `sum_all` can be instantiated with `int`.
2. `std::string`: it is not constructible from 0, so `sum_all` can not be instantiated with `std::string`.
3. `void`: it does not support any of the required operations, so `sum_all` can not be instantiated with `void`.
4. a pointer type: it does not support addition-assignment operation, so `sum_all` can not be instantiated with a pointer type. See `p3_test.cpp` for a demonstration of minimal class definition for successful instatiation of `sum_all`.

### Problem 4

The standard library provides a member function swap in the vector template for efficient exchange of vector content: instead of doing 3 copy operations, the swap function could be implemented as copy free by swapping the memory location of first elements in the vectors.

To take advantage of other exchange overloads, the overloaded `exchange` function for `triple` is written as a series of element-wise exchange operations:

```
template <typename T1, typename T2, typename T3>
void exchange(triple<T1, T2, T3>& x, triple<T1, T2, T3>& y) {
    cout << "using triple exchange: " << x << " <-> " << y << en
dl;
    exchange(x.first, y.first);
    exchange(x.second, y.second);
    exchange(x.third, y.third);
}
```

The test program is included in `p4_test.cpp`.