Homework #7 – `IntegerRange::iterator`

In this homework you are asked to implement an iterator for class template `IntegerRange`.

An IntegerRange represents all values within the range `[low, high)`. Internally the `IntegerRange` represents the values by only storing the low and high bounds; the actual values in the range are not generated. It will be the job of the iterator to produce the numbers in the range during iteration.

Here is the definition of `IntegerRange` *(you must not modify this definition)*:

```
// Represents all integer values in the range [low, high).
// low must be <= high.
template <typename T>
class IntegerRange
{
public:
   class iterator;

   IntegerRange(T low, T high) : low_(low), high_(high)
   {
      assert(low <= high);
   }

   const iterator begin() const { return iterator(low_); }
   const iterator end()   const { return iterator(high_); }

private:
   const T low_, high_;
};
```

Here are examples demonstrating how `IntegerRange` and its `iterator` might be used:

```
IntegerRange<int> r1(-2, 3);
copy(r1.begin(), r1.end(), ostream_iterator<int>(cout, " "));      // -2 -1 0 1 2

IntegerRange<unsigned> r2(0, 6);
copy(r2.begin(), r2.end(), ostream_iterator<unsigned>(cout, " ")); // 0 1 2 3 4 5
```

1. **(5 points)** Implement class template `IntegerRange<T>::iterator`. *Your implementation must support all operations required of the iterator category you decide is most appropriate.*
2. **(4 points)** Implement 4 tests demonstrating `IntegerRange<T>` working with 4 integer types (e.g. `int`, `long`) and 4 algorithms of your choice. *Note, each test is only required to test one integer type and one algorithm.*
3. **(1 point)** Make sure your source code is well-commented, consistently formatted, uses no magic numbers/values, follows a consistent style, and is ANSI-compliant.

Place all source code and a screen capture of the output produced by your program in a single PDF document. Submit this document.