C++11 STL additions

**bool all_of(Iter first, Iter last, Pred pred)**
true if all the values in [first, last) satisfy the predicate (or if the range is empty), false otherwise

**bool any_of(Iter first, Iter last, Pred pred)**
true if at least one of the values in [first, last) satisfies the predicate, false otherwise (or if the range is empty)

**bool none_of(Iter first, Iter last, Pred pred)**
true if no values in [first, last) satisfy the predicate (or if the range is empty), false otherwise

**Iter find_if_not(Iter first, Iter last, Pred pred)**
returns the first iterator i in the range [first, last) such that pred(*i) == false or last if no such iterator found

**Outiter copy_if(Iter first, Iter last, Outiter result, Pred pred)**
copies all elements in [first, last) that satisfy a predicate into a range starting from result (the opposite of remove_copy_if)

**Outiter copy_n(Iter first, Size n, Outiter result)**
copies n elements starting from first into a range starting from result

** uninitialized_copy_n(InIter first, Size n, Outiter result)**
invokes uninitialized_copy for n elements

**Iter move_backwarkd(Iter first, Iter last, Outiter result)**
moves elements from [first, last] into a range starting from result

**is_partitioned(Iter first, Iter last, Pred pred)**
true if [first, last) is empty or if [first, last) is partitioned by pred, i.e. if all elements that satisfy pred appear before those that don’t

**pair<Outiter1, Outiter2> partition_copy(Iter first, Iter last, Outiter1 out_true, Outiter2 out_false, Pred pred)**
copies elements that satisfy pred from [first, last) into the result range with out_true, and other elements into the range starting with out_false

**Iter partition_point(Iter first, Iter last, Pred pred)**
returns an iterator to the 1st element in [first, last) that doesn’t satisfy pred

**Raler partial_sort_copy(Iter first, Iter last, Raler result_first, Raler result_last)**
copies sorted elements from [first, last) into the result range (in terms of comp if supplied); the number of elements copied is determined by the size of the smaller of input and result ranges

**bool is_sorted(Iter first, Iter last)**
true if [first, last) is sorted (in terms of comp if supplied), false otherwise

**Iter is_sorted_until(Iter first, Iter last)**
returns the last iterator i in [first, last) for which the range [first, i) is sorted (in terms of comp if supplied)

**bool is_heap(Iter first, Iter last)**
true if [first, last) is a heap (in terms of comp if supplied), i.e. the first element is the largest

**Iter is_heap_until(Iter first, Iter last)**
returns the last iterator i in [first, last) for which the range [first, i) is a heap (in terms of comp if supplied)

**T min(Initializer_list<T> t)**
returns the smallest value (in terms of comp if supplied) in the initializer_list

**T max(Initializer_list<T> t)**
returns the largest value in the initializer_list (in terms of comp if supplied)

**pair<Const T a, Const T b> minmax(Initializer_list<T> t)**
returns the pair (b, a) if b < a (in terms of comp if supplied), and (a, b) otherwise

**array<T, N> stores fixed size sequences of objects (N elements of type T); elements are stored contiguously**

**unordered_set<T> contains at most one of each value and provides fast retrieval of values; supports forward iterators**

**unordered_multiset<T> supports equivalent values (possibly with multiple copies of the same value) and provides fast retrieval of the values; supports forward iterators**

**unordered_map<Key, T> hash table; contains at most one of each key value; supports forward iterators**

**unordered_multimap<Key, T>**
hash table; supports equivalent keys (can contain multiple copies of each key value); supports forward iterators

**unordered_set<T> contains at most one of each value and provides fast retrieval of values; supports forward iterators**

**unordered_multiset<T> supports equivalent values (possibly with multiple copies of the same value) and provides fast retrieval of the values; supports forward iterators**