Add `std::is_partitioned_until` algorithm

Document #:	P0000
Date:	2019-09-06
Project:	Programming Language C++
Audience:	LEWGI

Reply-to: Alexander Zaitsev <zamazan4ik@tut.by, zamazan4ik@gmail.com>

1 Revision history

- R0 – Initial draft

2 Motivation

`std::is_partitioned` was added long time ago to the standard library. The algorithm is useful but sometimes we need an infromation about "where a partition is broken". `std::is_partitioned` returns only `bool` and we cannot change an interface of existing function. So we can add additional function std::is_partitioned_until which returns an iterator instead of bool. `partition_point` cannot be used here because it works only on partitioned ranges.

3 Proposed wording

Add to [**alg.partitions**] **25.7.4**:

[...]

```
template<class InputIterator, class Predicate>
  constexpr InputIterator is_partitioned_until(InputIterator first, InputIterator last,
                                               Predicate pred);
template<class ExecutionPolicy, class InputIterator, class Predicate>
  InputIterator is_partitioned_until(ExecutionPolicy&& exec, InputIterator first,
                                     InputIterator last, Predicate pred);

template<input_iterator I, sentinel_for<I> S, class Proj = identity,
         indirect_unary_predicate<projected<I, Proj>> Pred>
  constexpr input_iterator ranges::is_partitioned_until(I first, S last, Pred pred,
                                                        Proj proj = {});
template<input_range R, class Proj = identity,
         indirect_unary_predicate<projected<iterator_t<R>, Proj>> Pred>
  constexpr iterator_t<R> ranges::is_partitioned_until(R&& r, Pred pred, Proj proj = {});
```

Let proj be identity for the overloads with no parameter named proj.

Returns: the last iterator `it` in the sequence [first, last) for which the `is_partitioned(first, it)` is true.

Complexity: Linear. At most last - first applications of pred and proj. [...]

4 Examples

Given the container c containing `0,1,2,3,14,15`, then

```
bool isOdd ( int i ) { return i % 2 == 1; }
bool lessThan10 ( int i ) { return i < 10; }

is_partitioned_until ( c, isOdd ) // iterator to '1'
is_partitioned_until ( c, lessThan10 ) // end
is_partitioned_until ( c.begin (), c.end (), lessThan10 ) // end
is_partitioned_until ( c.begin (), c.begin () + 3, lessThan10 ) // end
is_partitioned_until ( c.end (), c.end (), isOdd ) // end, because of empty range
```

5 Possible implementation

Also possible implementation can be found in Boost.Algorithm: GitHub. Documentation can be found here: Boost. Available in Boost.Algorithm since Boost 1.65.