# Add std::is_partitioned_until algorithm

Document #: P0000
Date: 2019-08-16
Project: Programming Language C++
Audience: LEWGI
Reply-to: Alexander Zaitsev <zamazan4ik@tut.by, zamazan4ik@gmail.com>

## 1 Revision history

- R0 – Initial draft

## 2 Abstract

## 3 Motivation

std::is_partitioned was added long time ago to the standard library. Algorithm is useful but sometimes we additionally need an infromation about "where partition is over". Current 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.

## 4 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 bool ranges::is_partitioned(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 InputIterator is_partitioned_until(R&& r, Pred pred, Proj proj = {});
```

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

Returns: true if and only if the elements e of [first, last) are partitioned with respect to the expression bool(invoke(pred, invoke(proj, e))).

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

## 5 Implementation

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