Partition

This algorithm allows us to divide or split ranges into separate sets.

> **ℹ What is a partition?**
> A partition of a set is a decomposition of a set in subsets so that each element of the set is precisely in one subset. The subsets are defined in C++ by a unary predicate so that the members of the first subset fulfill the predicate. The remaining elements are in the second subset.

C++ offers a few functions for dealing with partitions. All of them need a unary predicate `pre`. `std::partition` and `std::stable_partition` partition a range and returns the partition point. With `std::partition_point` you can get the partition point of a partition. Afterwards you can check the partition with `std::is_partitioned` or copy it with `std::partition_copy`.

Checks if the range is partitioned:

```
bool is_partitioned(InpIt first, InpIt last, UnPre pre)
bool is_partitioned(ExePol pol, FwdIt first, FwdIt last, UnPre pre)
```

Partitions the range:

```
FwdIt partition(FwdIt first, FwdIt last, UnPre pre)
FwdIt partition(ExePol pol, FwdIt first, FwdIt last, UnPre pre)
```

Partitions the range stable:

```
BiIt stable_partition(FwdIt first, FwdIt last, UnPre pre)
BiIt stable_partition(ExePol pol, FwdIt first, FwdIt last, UnPre pre)
```

Copies a partition in two ranges:

```
pair<OutIt1, OutIt2> partition_copy(InIt first, InIt last, OutIt1 result_true, OutIt2 result_
pair<FwdIt1, FwdIt2> partition_copy(ExePol pol, FwdIt1 first, FwdIt1 last, FwdIt2 result_true
```

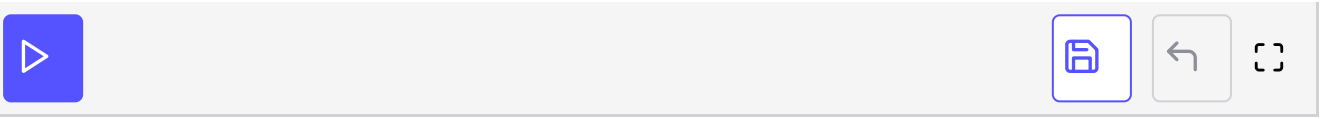
Returns the partition point:

```
FwdIt partition_point(FwdIt first, FwdIt last, UnPre pre)
```

A `std::stable_partition` guarantees, in contrary to a `std::partition`, that the elements preserve their relative order. The returned iterator `FwdIt` and `BiIt` points to the initial position in the second subset of the partition. The pair `std::pair<OutIt, OutIt>` of the algorithm `std::partition_copy` contains the end iterator of the subsets `result_true` and `result_false`. The behavior of `std::partition_point` is undefined, if the range is not partitioned.

```
#include <algorithm>
#include <cctype>
#include <deque>
#include <iostream>
#include <list>
#include <string>
#include <vector>

bool isOdd(int i){ return (i%2); }

int main(){

  std::cout << std::boolalpha << std::endl;

  std::vector<int> vec{1, 4, 3, 4, 5, 6, 7, 3, 4, 5, 6, 0, 4, 8, 4, 6, 6, 5, 8, 8, 3, 9, 3, 7

  for ( auto v: vec ) std::cout << v << " ";

  std::cout << "\n\n";

  auto parPoint= std::partition(vec.begin(), vec.end(), isOdd);

  for (auto v: vec) std::cout << v << " ";
  std::cout << std::endl;
  for (auto v= vec.begin(); v != parPoint; ++v) std::cout << *v << " ";
  std::cout << std::endl;
  for (auto v= parPoint; v != vec.end(); ++v) std::cout << *v << " ";
  std::cout << std::endl;

  std::cout << std::endl;

  std::cout << "std::is_partitioned: " << std::is_partitioned(vec.begin(), vec.end(), isOdd)

  std::cout << "std::partition_point: " << (std::partition_point(vec.begin(), vec.end(), isOd

  std::cout << std::endl;

  std::list<int> li;
  std::list<int> de;
  std::partition_copy(vec.begin(), vec.end(), std::back_inserter(li), std::back_inserter(de),
```

```
  for (auto v: li) std::cout << v << " ";
  std::cout <<  std::endl;
  for (auto v: de) std::cout << v << " ";

  std::cout << "\n\n";

}
```



Partition algorithms