Is it copied ?

```cpp
struct Data {
    std::list<int> numbers;
};

template <class UnaryPredicate>
std::vector<Data> filter_data(std::vector<Data> && lots_of_data, UnaryPredicate && f)
{
    auto const begin = std::move_iterator(lots_of_data.begin());
    auto const end   = std::move_iterator(lots_of_data.end());

    auto filtered = std::vector<Data>{};
    std::copy_if(begin, end, std::back_inserter(filtered), FWD(f));
    return filtered;
}
```

Is there Data copies ?

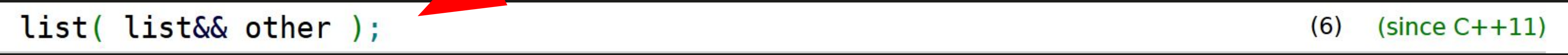
```
Is there Data copies ?
    GCC : 0
```

```
Is there Data copies ?
    GCC : 0
    Clang : 0
```

```
Is there Data copies ?
    GCC : 0
    Clang : 0
    MSVC : ~2.5 * filtered.size()
```

On cppreference :

noexcept pls



```
list( list&& other );
```

(6) (since C++11)

```cpp
BigThing create_thing(bool b) {
    if (!b) return {};

    auto const thing = BigThing{ 1, 2, 3 };
    log("Success !");
    return thing;
}
```

```cpp
BigThing create_thing(bool b) {
    if (!b) return {};

    auto thing = BigThing{ 1, 2, 3 };
    log("Success !");
    return thing;
}
```

```cpp
std::pair<BigThing, std::error_code> big_pair() {
    auto big = BigThing{};
    return { big, {} };
}
```

```cpp
std::pair<BigThing, std::error_code> big_pair() {
    auto big = BigThing{};
    return { std::move(big), {} };
}
```

```cpp
std::pair<BigThing, std::error_code> big_pair() {
    auto big = BigThing{};
    return { std::move(big), std::error_code{} };
}
```

```cpp
template <class T>
decltype(auto) move(T&& value) {
   using value_type = std::remove_reference_t<T>;

   static_assert(!std::is_const_v<value_type>);
   static_assert(std::is_nothrow_move_constructible_v<value_type>);
   static_assert(std::is_nothrow_move_assignable_v<value_type>);

   return static_cast<value_type&&>(value);
}
```

Nobody throws :

std::forward_list, std::array (*), std::vector (so default std::priority_queue as well)

MSVC throws :

std::list, std::function, vector<bool>, std::set & std::map (+ unordered_x and multi_x)

Everybody throws :

std::deque (so default std::stack and std::queue as well)

“This is ugly !”

- *The bad programmer*

“This is ugly !”

- *The good programmer*