Using Searchers

The lesson will elaborate the syntax and methods for using Searchers.

The `std::search` function uses the following overload for searchers:

```
template<class ForwardIterator, class Searcher>
ForwardIterator search(ForwardIterator first, ForwardIterator last, const Searcher& searcher
```

For example:

```
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;
int main(){
    string testString = "Hello Super World";
    string needle = "Super";
    const auto it = search(begin(testString), end(testString), boyer_moore_searcher(begin(nee

    if (it == cend(testString))
        cout << "The string " << needle << " not found\n";
    return 0;
}
```

Each searcher initializes its state through the constructor. The constructors need to store the pattern range and also perform the preprocessing phase. Then `std::search` calls their `operator()(iter TextFirst, iter TextLast)` method to perform the search in the text range.

Since a searcher is an object, you can pass it around in the application. That might be useful if you'd like to search for the same pattern inside various text objects. In that case, the preprocessing phase will be done only once.

Enough of the syntax!
Let's go through some examples in the next lesson!

Let's go through some examples in the next lesson.