Exercise #1 - findNeedles

This section contains a short code sample written in C++.

This exercise has two required parts:

1. Write an API reference document that explains how to call this method. Your audience for this document is an experienced C++ programmer.
2. Assume you have a chance to send comments or questions to the person who wrote the code. Suggest ways to improve the code, for example, to reduce memory usage or enhance features.

```
#include <iostream>
#include <iterator>
#include <sstream>
#include <string>
#include <vector>

void findNeedles(const std::string &haystack,
                          const std::vector<std::string> &needles) {
  if (needles.size() > 5) {
    std::cerr << "too many words!";
  }

  else{
    int countArray[needles.size()];
    for(int j = 0; j < needles.size(); j++){
      countArray[j] = 0;
    }

    std::istringstream haystack_ss(haystack);
    std::vector<std::string> words;

    // Split haystack into words
         std::copy(std::istream_iterator<std::string>(haystack_ss),
                   std::istream_iterator<std::string>(),
                       std::back_inserter<std::vector<std::string> >(words));

    for(int i = 0; i < needles.size(); i++) {
      for(int j = 0; j < words.size(); j++){
        if(words[j] == needles[i]){
          countArray[i]++;
        }
      }
```

```
    }

    for(int j = 0; j < needles.size(); j++){
      std::cout << needles[j] << ": " << countArray[j] << std::endl; }
  }
}
```

findNeedles

Description

The method **findNeedles** takes in two parameters (**haystack** and **needles)** and checks to see how many **needles** are in the **haystack**. **Haystack** is a constant string and **needles** is a constant vector of strings.

The size of **needles** cannot be greater than 5, otherwise, an error message is printed. If the vector, **needles**, contains 5 items or less, the function counts each item's occurrence in the string, **haystack**.

To count the occurrence of each **needle** in the **haystack**, **haystack** is first split into a vector of strings, called **words**. Then the function checks whether the first item in the list, **needles**, appears in **words**. If it does not, then the same **needle** is compared to the next item in **words**. Once that item in **needles** has been compared to all the **words**, the next **needle** is compared to each item in **words**.

For every time an item in **needles** occurs in the **words** list, the corresponding **countArray** value is increased by 1. Therefore, **countArray** keeps a tally of the total number of occurrences of each **needle**. Once every **needle** in the list is compared, it is displayed along with the number of times it occurred in the **haystack**.

Parameters

The function **findNeedles** has two parameters, a string of word(s) (called **haystack**) and a vector of strings (called **needles**).

Return Values

The method **findNeedles** does not return anything but prints the output.

Function call

FindNeedles is a function that accepts two parameters: **haystack** and **needles**. To call this function, you must assign a string to each parameter within the main function. The data type for these parameters are **std::string** and **std::vector<std::string>**, respectively.

Example 1: length of needles is equal 5

For example, **haystack** can be assigned the following string:

```
std::string haystack = "This is a Google interview question. It is interesting.";
```

While **needles** can be assigned the following string:

```
std::vector<std::string> needles = {"This", "is", "a", "Google", "campus"};
```

Finally, the function must be called with each parameter:

```
findNeedles (haystack, needles);
```

The output of the example should be:

```
This: 1
is: 2
a: 1
Google: 1
campus: 0
```

Example 2: length of needles is greater than 5

For example, **haystack** can be assigned the following string:

```
std::string haystack = “This is a Google interview question.”;
```

While **needles** can be assigned the following string:

```
std::vector<std::string> needles = {“This”, “is”, “a”, “Google”, “campus”, “interview”};
```

Finally, the function must be called with each parameter:

```
findNeedles (haystack, needles);
```

The output of the example should be:

```
too many words!
```

Code Optimization

The code can be optimized by including the header “using namespace std;”, which will remove all the “std” phrases throughout the code. Additionally, the third for loop can be removed, and the output of each needle in the haystack can be printed immediately within the second for loop, as shown below.

```
for(int i = 0; i < needles.size(); i++) {
  for(int j = 0; j < words.size(); j++){
    if(words[j] == needles[i]){
      countArray[i]++;
    }
   }
  cout << needles[i] << ": " << countArray << endl;
 }
```