C++ code kata: Week #5

Hello 👋
First of all congratulations to you only completing four weeks of code kata. Hope, your journey up to now is plausible.

An **Anagram** is a rearrangement of words or sentences to produce a new word or sentence.

Fig. 1 Anagram of word *Listen*.

Ground Exercise

In today's exercise 🏋 you have to write a function `is_anagram()` that will take **two strings** as arguments and returns if argument 1 is an anagram of argument 2.

The example below shows how we use `is_anagram()`. e.g. 👇

```
auto result = is_anagram("listen", "silent");
# result will contain
# true
```

Rules

1. You have to pass two strings only.
2. Anagrams are case insensitive.
3. The return value will be either true or False.
4. You cannot use any library other than the C/ C++ standard library.

Tools

To write solutions you can download tools from:

Tool	Usage	Download Location
Visual Studio Code	Lightweight program editor.	Download
GCC (Linux)	C / C++ Compiler.	Download
MingGW (Windows)	C / C++ Compiler.	Download
XCode toolkit (MacOS)	IDE with multi language support.	Download
Catch2	C++ Test framework	Download

Test Script

Test Script for this week's problem can be downloaded from **here**.

Sensei Says

"What you learn is not what you read or listened to, but rather what you attempted at..."

Progressive learning

If you feel the exercise a little bit difficult to solve do not get disheartened. The whole idea behind these programming exercises is not to solve them but rather attempt them.

Try to attempt them in as many ways as possible, you will learn new techniques that will be very helpful to you in the long run especially in the work field.

We will present you with **solution mail/document** also. The solution will show you various ways to solve a problem and why a technique is better than the last one.

We encourage you to make notes from the solution provided and try to apply what you have learnt in the future exercise.

What I will gain from these exercises ?

1. A better and faster way to solve the exercise.

2. Reusable components like containers, algorithms etc. that you can apply to the problem at hand rather than designing your own.
3. Confidence and attitude to solve a problem in new ways, instead of trying monotonous techniques.
4. Writing the robust and quality software using **test driven development**.

All the best 🤞

In case you have any difficulty using the tools or some questions related to this exercise. Reach us at

1. 💬 programmingdays (Skype)