CCPS506 Lab 7 – Rust: Functions and Ownership

Preamble

In this lab you will experiment with writing some simple Rust functions. This would be CCPS109-level stuff in Python but passing arguments and returning values in Rust becomes a lot trickier due to issues of ownership and strict typing.

Lab Description

Implement each of the following Rust functions in the single file called **`Lab7.rs`**

i) Write a Rust function called **`stringConcat`** that accepts two string slices as input and returns a new String that is the concatenation of the two.

ii) Write a Rust function called **`subArrayAverage`** that accepts an array of floating-point values, and a tuple containing an upper and lower bound. Return the average of the values in the sub-array defined by the upper and lower bounds. The lower bound is inclusive, the upper bound is not. If the tuple does not contain valid bounds (negative numbers, index out of bounds, etc.) simply return zero.

iii) Write a Rust function called **`arraySignum`**. It should accept an array of integers, and return a **vector** containing the values -1, 0, or 1 depending on the sign of the elements in the argument array. Your function should not modify the input array, but rather create and return a new vector.

Since we may not have covered Vectors in class, this is a good opportunity to practice some independent learning. This is good practice if you wish to complete the assignment in Rust as it requires working with vectors. Vectors in Rust are a lot like vectors in C++. Check out the documentation for more:

`https://doc.rust-lang.org/std/vec/struct.Vec.html`

In addition to these three functions, your **`Lab7.rs`** file should contain a main function that calls each of the above functions with an example or two to make sure they work correctly.

Submission

Labs are to be completed and submitted *individually*. Submit your **Lab7.rs** source file containing all completed functions on D2L, under the submission for Lab 7.