CCPS506 Lab 8 – Rust: Custom types and methods

Preamble

In this lab, you will practice defining a custom type in Rust, and implementing methods over that type. You will also get some experience using HashMap, which is just like its Java counterpart, and is like a dictionary in Python.

Lab Description

Create a Rust program called **`Lab8.rs`** that implements a polynomial type as a struct, called **`Poly`**. The polynomial struct will have two fields:

1) A vector containing the coefficients of the polynomial. The type should be **`i32`**.

2) A HashMap containing the results for inputs that the polynomial has been evaluated over. The type of the HashMap should be **`<i32, i32>`**. Documentation for HashMap in Rust can be found at the following link:

   https://doc.rust-lang.org/std/collections/struct.HashMap.html

   If **`f(x)`** is the result of evaluating the polynomial over **`x`**, then in the HashMap **`x`** would be the key and **`f(x)`** would be the value at that key. This means that when the polynomial is created, the HashMap is empty. As the Polynomial is evaluated over different values of x, the HashMap will grow. For the example below:

   $$f(x) = 2x^2 + 3x + 4$$
   $$f(2) = 2*2^2 + 3*2 + 4 = 18$$

   The coefficient vector would contain 2, 3, 4. After evaluating **`f(2)`**, the HashMap would contain the value 18 at the key 2.

You will implement a method for your **`Poly`** type called **`eval`** that evaluates the polynomial for a provided value. I.e. **`p1.eval(2)`** should evaluate polynomial **`p1`** at **`x = 2`** and return the result. Before evaluating the polynomial, you should check the HashMap to see if the polynomial has already been evaluated for that input. If it has, return the value from the HashMap. If it hasn't, compute the result, add it to the HashMap, and then return the result.

Submission

Labs are to be completed and submitted *individually*. Submit your **Lab8.rs** source file containing all completed functionality on D2L, under the submission for Lab 8.