Institut für Informatik **Ludwig-Maximilians-Universität München**
Prof. Dr. Francois Bry

Higher level languages: Rust, WS 18/19
Tutorial 3

October 31, 2018

Exercise 3-1 Roots of polynomials using Newton's method

a) Create a custom data type for the representation of polynomials

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_2 x^2 + a_1 x + a_0,$$

storing the coefficients a_i and the exponents i.

Because we want to represent polynomials of arbitrary length the data type must be recursive. Because of this, subtasks b, c, and d should be also implemented in a recursive fashion. `Vec` and similar pre-defined composite data types should not be used for this exercise.

b) Implement a method for printing polynomials. To do this, visit every single term of the polynomial and append its textual representation to an initially empty value of type `String`. When done, return this string value to the program's main function which should output it to the screen.

c) Implement a method for evaluating a polynomial for a particular value of x. For raising a particular value to an integer exponent you can use the method `fn powi(&self, n: i32) -> f32`. Documentation and an example can be found at `https://doc.rust-lang.org/std/primitive.f32.html#method.powi`

d) Implement a method for differentiating a polynomial. A possible type signature for differentiation could look like this: `fn differentiate(&self) -> Poly`

e) Implement a method that uses Newton's method to find a root of a polynomial. See the description below.

Most likely you will need to convert between data types. This can be achieved with the explicit type cast syntax:

```
let i: i32 = 5;
let f: f32 = i as f32;
```

Newton's Method

Newton's method is a way for finding approximations to the roots of functions. The process starts with an initial guess x_0 and uses the function f and its derivative f' in order to close the gap between the estimates x_i and the actual solution:

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

The algorithm should terminate when it reaches a certain precision, i.e. the difference between two consecutive guesses is below a certain threshold.

Test your solution

You could use the following polynomials in order to check the validity of your program:

polynomial	initial guesses	roots
$p(x) = x^3 - 2x^2 - 11x + 12$	$-4, 0, 2.35287527$	$-3, 1, 4$
$p(x) = x^3 - 2x^2 - 5x + 6$	$-3, 0, 4$	$-2, 1, 3$
$p(x) = 2x^4 + 7x^3 + 6x^2 + 8x + 12$	$0, 2.5$	$-1.5, -2.5943$