RUST Launch Pad

Worksheet # 3

1. Create a Rust function that takes a sentence as input, splits it into words, and returns a vector of unique words in alphabetical order.
2. Write a Rust program that takes a vector of integers as input, squares each element, and then calculates the sum of the squared values.
3. Implement a program that reads a text file and counts the frequency of each word using a HashMap. Ignore punctuation and consider words in a case-insensitive manner.
4. Write a Rust function that reads a text file specified by its path and returns the content as a String. Handle errors using the Result type.
5. Create a function that parses a string as an integer. If the parsing fails, return an Option with None; otherwise, return Some(parsed_integer).
6. Implement a program that reads a configuration file and returns a custom error type if the file format is invalid. Use the ? operator for error propagation.
7. Create a generic function that filters elements of a vector based on a provided predicate function. The function should work for vectors of different types.
8. Define a trait called `Drawable` with a method `draw`. Create a struct called `Shape` that stores the name of a shape and implement the `Drawable` trait for it.
9. Design a geometry library in Rust. Create traits like `Area` and `Perimeter` that have methods for calculating area and perimeter for various geometric shapes (e.g., Circle, Rectangle). Implement these traits for the respective struct types.
10. Create a function that takes a sentence as input and converts it to "Title Case" (capitalize the first letter of each word) while ignoring common articles and prepositions like "the," "in," "of," etc.
11. Write a Rust program that takes user input and appends it to a text file specified by its path. Handle errors using the Result type and ensure the file is properly closed.
12. Implement a function that takes a HashMap containing student names and their scores as input. Return a new HashMap with only the students who scored above a certain threshold.
13. Create a function that divides two integers and returns a Result with the result of the division if the denominator is not zero. If the denominator is zero, return an error indicating division by zero.
14. Implement a program that reads data from a CSV file and converts it into a custom data structure. Define custom error types for various potential parsing errors and use the ? operator for error handling.

15. Design a sorting library in Rust that provides a trait called `Sortable` with a method `sort` for sorting arrays or vectors. Implement this trait for various data types and sorting algorithms (e.g., Bubble Sort, Quick Sort).