Student Name: _____ Student ID: _____ TA: _____

Rust Lab 11 – Ranges, Slices and Iterators

17/9/2025

Objectives:

Practice defining and using ranges and slices in functions. Understand differences between arrays, slices, and vectors. Implement and use iterators, including custom ones.

Lab 1: Creating Custom Iterators

The Fibonacci sequence is defined as F(n) = F(n-1) + F(n-2), with F(1) = 0 and F(2) = 1 (starting: 0, 1, 1, 2, 3, 5, ...).

Requirements:

1. Create a struct named Fibonacci with fields for the current and next numbers.
2. Implement the Iterator trait for Fibonacci. The next() method should:
   - Return the current Fibonacci number.
   - Update the state to generate the next number.
3. Implement a constructor (new()) for Fibonacci that initializes with 0 and 1.
4. In the main function:
   - Use the iterator to print the first 10 Fibonacci numbers.
   - Format as "Fibonacci N: value", using 1-based indexing for N.

CODE Example

```
struct Fibonacci {
    current: u64,
    next: u64,
}

impl Fibonacci {
    fn new() -> Fibonacci {
        Fibonacci { current: 0, next: 1 }
    }
}

impl Iterator for Fibonacci {
    type Item = u64;

    fn next(&mut self) -> Option<Self::Item> {
    }
}

fn main() {
    for (i, val) in fib.take(10).enumerate() {
    }
}
```

Expected Output:

```
Fibonacci 1: 0
Fibonacci 2: 1
Fibonacci 3: 1
Fibonacci 4: 2
Fibonacci 5: 3
Fibonacci 6: 5
Fibonacci 7: 8
Fibonacci 8: 13
```

Hint: Use u64 for fields to avoid overflow in early terms.

TA Check: _____

Lab 2: Weather Data Analysis

Scenario: Analyze a 30-day weather dataset with daily average temperature (f64), humidity (u32), and rain status (bool).

Task:

1. Warmest Period: Write warmest_period to find the consecutive 3 days with the highest average temperature. Input: weather data. Output: slice of the warmest period. If multiple have the same average, select the earliest group. Example with temperatures [1.0, 3.0, 5.0, 2.0, 9.0, 10.0, 4.0, 7.0, 6.0, 3.0]:

Window	Days	Average
0-2	1, 3, 5	3.0
1-3	3, 5, 2	3.33
2-4	5, 2, 9	5.33
3-5	2, 9, 10	7.0
4-6	9, 10, 4	7.67
5-7	10, 4, 7	7.0
6-8	4, 7, 6	5.67
7-9	7, 6, 3	5.33

Highest: [9, 10, 4] with 7.67.

2. Coldest Day: Write coldest_day to find the 0-based index of the day with the lowest temperature. If ties, return the earliest index.
3. Rain Prediction: Write predict_rain to predict rain on a day: true if 0.005 * humidity + 0.02 * temperature - 1 > 0.5.
4. Rainy Days: Write count_rainy_days to count days where it rained (true).

Data:

let weather_data = vec![(25.0, 65, false), (26.2, 70, false), (24.8, 62, false), (23.5, 78, true), (22.1, 82, true), (20.7, 85, true), (21.3, 80, true), (22.8, 73, false), (24.0, 68, false), (25.5, 60, false), (27.1, 55, false), (28.3, 52, false), (27.9, 58, false), (26.6, 64, false), (25.2, 70, true), (23.8, 75, true), (22.4, 80, true), (21.0, 83, true), (20.5, 86, true), (21.8, 82, true), (23.2, 77, false), (24.5, 70, false), (25.8, 63, false), (26.4, 58, false), (27.0, 53, false), (26.7, 56, false), (25.3, 62, false), (24.9, 68, true), (23.1, 74, true), (21.7, 79, true)];

Expected Output:

```
Warmest 3-day period: [(27.1, 55, false), (28.3, 52, false), (27.9, 58, false)]
Coldest day: 18
Number of rainy days: 13
Will it rain on the first day? false
```

Hints:

- Utilize slicing and ranges to extract relevant portions of the weather_data.
- Employ iteration and conditional logic to analyze the data and make predictions.
- Use Rust's iterator methods (e.g., iter(), map(), filter(), windows()) for concise and efficient code.

CODE Example

```
fn warmest_period(data: &[WeatherData]) -> &[WeatherData] {
    data.windows(3)
        .max_by(|a, b| {})
        .unwrap_or(&[])
}
fn coldest_day(data: &[WeatherData]) -> usize {
    data.iter()
}
fn predict_rain(day_data: &WeatherData) -> bool {
}
fn count_rainy_days(data: &[WeatherData]) -> usize {
}
fn main() {
    let weather_data: Vec<WeatherData> = vec![
    ];
    println!("\n--- Lab 2 Output ---");
    // Expected Output formatting. [cite: 36, 37, 38, 39]
    println!("Warmest 3-day period: {:?}", warmest_period(&weather_data));
    println!("Coldest day: {}", coldest_day(&weather_data));
    println!("Number of rainy days: {}", count_rainy_days(&weather_data));
    println!("Will it rain on the first day? {}", predict_rain(&weather_data[0]));
}
```

TA Check: _____