Day 1 - Closure Exercise

Exercise 1: Basic Closure

Task: Create a closure that:

1. Takes one integer parameter
2. Returns the value multiplied by 3
3. Call it with numbers 5 and -2

Rust

```rust
fn main() {
    // Create your closure here
    // let triple = ...;

    println!("5 tripled: {}", triple(5));
    println!("-2 tripled: {}", triple(-2));
}
// Expected output: 15, -6

```

Exercise 2: Closure Environment Capture

Task: Create a closure that:

1. Captures a value `n` from its environment
2. Takes one parameter `x`
3. Returns `x + n`
4. Test with different values of `n`

Rust

```rust
fn main() {
    let n = 10;
    // Create your closure here that captures n

```

```rust
    println!("5 + {} = {}", n, add_n(5));
    let n = -3;
    println!("5 + {} = {}", n, add_n(5));
}
// Expected output: 15, 2

```

Exercise 3: Closure as Function Parameter

Task: Create a function `apply_operation` that:

1. Takes a closure and an integer
2. Applies the closure to the integer
3. Test with both addition and multiplication closures

Rust

```rust
fn apply_operation(f: impl Fn(i32) -> i32, x: i32) -> i32 {
    // Implement function body
}

fn main() {
    let add_five = |x| x + 5;
    let double = |x| x * 2;

    println!("{}", apply_operation(add_five, 10)); // 15
    println!("{}", apply_operation(double, 10));   // 20
}

```

Exercise 4: Closure in Iterator

Task: Use closures with iterator methods to:

1. Take a range 1..=20
2. Filter even numbers
3. Square remaining numbers
4. Sum the results

```rust
fn main() {
    let result = (1..=20)
        // Add iterator methods with closures here
        .sum::<i32>();

    println!("Sum of squares of even numbers: {}", result);
}
// Expected output: 1540 (sum of 4+16+36+64+100+144+196+256+324+400)

```

Bonus Exercise: FnOnce, FnMut, Fn Traits

Task: Create three closures demonstrating:

1. A closure that can only be called once (FnOnce)
2. A closure that modifies its environment (FnMut)
3. A closure that only reads its environment (Fn)

```
fn main() {
    // FnOnce example (consumes a value)
    let name = "Alice".to_string();
    let greet_once = move || {
        println!("Hello {}", name);
        name // Return name, consuming it
    };
    greet_once();
    // greet_once(); // This should fail

    // FnMut example (modifies counter)
    let mut counter = 0;
    let mut increment = || {
        counter += 1;
        println!("Counter: {}", counter);
    };
    increment();
    increment();

    // Fn example (reads but doesn't modify)
    let read_only = |x| x + counter;
    println!("Read: {}", read_only(5));
}
```

```
24
```

Each exercise focuses on different closure aspects:

1. Basic syntax and calling
2. Environment capture
3. Passing closures as parameters
4. Ownership and moving
5. Using with iterators

Bonus: Closure trait differences