Option Enum - Exercise

1. Safe Division

```rust
// Create a function that returns Some(a/b) if b ≠ 0, otherwise None
fn safe_divide(a: f64, b: f64) -> Option<f64> {
    todo!()
}

// Example test case:
// assert_eq!(safe_divide(10.0, 2.0), Some(5.0));
// assert_eq!(safe_divide(5.0, 0.0), None);

```

2. First Character of a String

```rust
// Write a function that returns Some(first character) if the string is non-empty, otherwise None
fn first_char(s: &str) -> Option<char> {
    todo!()
}

// Example test case:
// assert_eq!(first_char("hello"), Some('h'));
// assert_eq!(first_char(""), None);

```

3. Find User by ID

```rust
struct User {
    id: u32,
    name: String,
}

```

```rust
// Search a vector of Users and return Some(user) if id matches, otherwise None
fn find_user(users: Vec<User>, target_id: u32) -> Option<User> {
    todo!()
}

// Example test case:
// let users = vec![
//     User { id: 1, name: "Alice".to_string() },
//     User { id: 2, name: "Bob".to_string() }
// ];
// assert_eq!(find_user(users.clone(), 1).unwrap().name, "Alice");
// assert_eq!(find_user(users, 3), None);

```

4. Config Value Parser

Rust

```rust
// Parse a string config value into different types, returning None if parsing fails
fn parse_config_value(value: &str) -> Option<ConfigValue> {
    #[derive(Debug, PartialEq)]
    enum ConfigValue {
        Int(i32),
        Float(f64),
        Bool(bool),
        String(String),
    }
    todo!()
}

// Example test case:
// assert_eq!(parse_config_value("42"), Some(ConfigValue::Int(42)));
// assert_eq!(parse_config_value("true"), Some(ConfigValue::Bool(true)));
// assert_eq!(parse_config_value("3.14"), Some(ConfigValue::Float(3.14)));
// assert_eq!(parse_config_value("hello"), Some(ConfigValue::String("hello".to_string())));
// assert_eq!(parse_config_value("not_a_number"), None);

```