Exercise Generic Type and Traits:

Exercise 1:

Write a function `smallest` that returns the smallest of two values using generics.

Exercise 2:

- A generic API function that returns `Result<Response<Data>, Error>`
- Use a custom `Payload` struct as input
- Use custom names instead of `T` and `E`
- Simulate a mock API call that can succeed or fail

A. Marker Trait

Exercise 4: Create and Use a Marker Trait

Create a marker trait `Trackable`. Implement it for a struct `Vehicle`.

Then write a function that only accepts `Trackable` types.

B. Auto Trait

Exercise 5: Check if a Type is Send

Use the following function to test whether your struct is `Send`.

```
fn is_send<T: Send>() {}
```

Try it with both a safe struct and a struct containing raw pointers.

C. Unsafe Trait

Exercise 6: Declare and Implement an Unsafe Trait

Define an unsafe trait `SystemCall`. Implement it for a struct `SysOp`.

Use `unsafe` blocks properly to call this trait's method.

Exercise 7: Trait Bound with Multiple Traits

Write a generic function that works only if the type implements both `Debug` and `Clone`.

Exercise 8:

Create a `trait Summarize` with a `summary()` method. Implement it for a `NewsArticle and Tweet` struct.