Student Name: ______________________________ Student ID: ________________ TA: ______________________

Rust Lab 07 – Defining Generic Functions and Types

13/8/25

Lab 1: Generic Container Swapper Goal: Implement a simple generic function to swap the contents of two vectors.
Function Signature:

```
fn swap_elements<T>(a: Vec<T>, b: Vec<T>) -> (Vec<T>, Vec<T>)
```

Tasks:
1. Accept two vectors of type T.
2. Return a tuple with their elements swapped.

Example:

```
Input: [1, 2, 3], [4, 5, 6]
Output: [4, 5, 6], [1, 2, 3]
```

TA Checking: ____

Lab 2: Generic Data Storage System

Designing a flexible data storage system for a multi-purpose application.

Task:

1. Create a generic struct DataStore<T> that can hold a vector of items of type T.
2. Implement methods for DataStore<T>:
   - add_item(item: T)
   - remove_item(index: usize) -> Option<T>
   - get_item(index: usize) -> Option<&T>
   - find_item<F>(&self, predicate: F) -> Option<&T> where F is a closure F: Fn(&T) -> bool
3. Create an enum DataType<T> with variants Number(T), Text(String), and Boolean(bool).
4. Implement a print method for DataType<T> that formats the output based on the variant.
5. In the main function:

A) DataStore<DataType<i32>>

1. Create + add	Add: Number(42), Number(7), Number(128), Text("Rust".into()), Boolean(true)
2. Print count	Print len() and is_empty()
3. Get by index	get_item(0) → print with println!("{}", item)
	get_item(99) → handle None and print "None"
4. Find by closure	Find first number > 100:
	find_item(|x| matches!(x, DataType::Number(n) if *n > 100))
5. Remove	remove_item(1) (should remove Number(7)), print removed value (or None)
6. List all	Iterate for (i, it) in store.items.iter().enumerate() and println!("[A] {i}: {}", it)

B) DataStore<DataType<f64>>

1. Create + add	Add: Number(3.14), Number(2.71), Text("pi".into()), Boolean(false)
2. Get + Find	get_item(0) and print
	Find first number >= 3.0:
	find_item(|x| matches!(x, DataType::Number(v) if *v >= 3.0))
3. Remove tail	remove_item(store.len() - 1); print removed value
4. List all	Print all items as in A(6)

C) DataStore<String>

1. Create + add with String methods

- Build s = String::from("Hello"); s.push_str(" World"); s.push('!');
- Make s2 = format!("{} from Rust", s) (keep using s later)
- Add strings: s2, "functional".into(), "generics".into()

2. Get + Find
    - get_item(0) and print with println!("{:?}", item) (String)
    - Find string containing "Rust": find_item(|t| t.contains("Rust"))
3. Remove + modify + re-add
    - remove_item(0) → modify with .replace("World","โลก").to_uppercase() → add_item(modified)
4. List all	Print all strings with indices: println!("[C] {i}: {:?}", it)

Expected prints (pattern)

- Section headers like [A], [B], [C]
- len/empty line
- get(0) shows an item; get(99) shows None
- find(...) shows a matching item or None
- remove(...) shows Some(...) or None
- Final listing prints each item with index

TA Checking: ____

Lab 3: Generic Data Analysis Tool

Implement a simple analysis trait for numerical data.
Tasks:
1. Define trait SimpleAnalyzable with method mean(&self) -> f64.
2. Implement SimpleAnalyzable for Vec<f64>.
3. Create struct SimpleDataSet with Vec<f64> and implement SimpleAnalyzable.
4. Implement method filter<F>(&self, predicate: F) -> Self.
   where F: Fn(&f64) -> bool

Starter hint for mean:

```
  fn mean(&self) -> f64 {
    let sum: f64 = self.iter().sum();
    sum / self.len() as f64
  }
```

Example main:

```
  let data = SimpleDataSet::new(vec![1.0, 2.0, 3.0]);
  println!("Mean: {}", data.mean());
```

TA Checking: ____