Assignment: Practice Exercise on Ownership **Questions for this assignment**

1. Consider the code below. There are two inputs to the function of **some\_function()**. The first one is a String and the second one is a String slice. There is a compile time error in this program. Try to identify the error and then make suitable adjustments in the code so that the error is removed

fn main(){

let s1:String = String::from("this is me");

let s2: &str = "myself";

some\_function(s1,s2);

println!("{} {}",s1,s2); }

fn some\_function(a1: String, a2: &str){

println!("{} {}",a1,a2); }

1. Consider the simple program given below. Who will be the owner of the vec1 (defined inside the function, i.e., **recieving\_ownership\_from()**)in the main function when the function call completes.

fn main () {

let v1 = recieving\_ownership\_from();

}

fn recieving\_ownership\_from() -> Vec<i32>{

    let vec1 = vec![4,5,6,9];

    vec1

}

1. Consider the following program given below. Who owns the vector containing the values of [5, 8, 9, 7] at the end of the main function, when the function call to the function **recieving\_giving\_back\_ownership()** completes

fn main(){

    let v2 = vec![5, 8,9,7];

    let v3 = recieving\_giving\_back\_ownership(v2);

    }

fn recieving\_giving\_back\_ownership(vec1: Vec<i32>) -> Vec<i32>{

    vec1

}

1. Consider the program given below. What will be the value of the variable stack\_num at the end of the main function

fn main() {

    let mut stack\_num = 56;

    stack\_num  = stack\_function(stack\_num);

    println!("The value of the stack\_num after the updation is {} ", stack\_num);

    }

    fn stack\_function(mut var:i32) -> i32 {

    var = 50;

    var

    }

1. Given the following code, what is the issue with the borrowing?

fn main() {

    let mut some\_vec = vec![1, 2, 3];

    let first = get\_first\_element(&some\_vec);

    some\_vec.push(4);

    println!("The first number is: {}", first);

}

fn get\_first\_element(num\_vec: &Vec<i32>) -> i32 {

    num\_vec[0]

}

1. In the following code, what is the pontential issue with respect to borrowing ?

struct Person {

    name: String,

    age: i32,

}

fn main() {

    let mut person = Person {

        name: String::from("Alice"),

        age: 30,

    };

    change\_person(&person);

    println!("Name: {}", person.name);

}

fn change\_person(person: &Person) {

    person.name = String::from("Bob");

}

1. What is the issue with the borrowing in the code below?

fn main() {

    let mut numbers = vec![1, 2, 3];

    let slice = get\_slice(&mut numbers);

    numbers.push(4);

    println!("Slice: {:?}", slice);

}

fn get\_slice(numbers: &mut Vec<i32>) -> &[i32] {

    &numbers[0..2]

}