Assignment: Practice Exercise on Functions and Inputs

**Questions for this assignment**

1. Modify the program below by adding the definition of the functions "add()" so that it compiles correctly

fn main() {

    let x = (5 + 3) \* (6 + 4);

    let y = times(add\_3(5), add\_4(6));

    assert\_eq!(x, y);

    println!("Good job!");

}

1. Rewrite the main function in a way so that there is no variable in it and it performs the same job as this program. Your program should make calls to both the functions in this program.

fn double(x: i32) -> i32 {

    x \* 2

}

fn triple(x: i32) -> i32 {

    x \* 3

}

fn main() {

    let x = triple(double(5));

    let y = triple(x);

    println!("Answer: {}", y);

}

1. Write a function which will accept a tuple called point representing the x-axis and y-axis coordinates of a point. The function will compute the distance of the point from the origin and will return the computed distance.

The template of the function is given below

fn print\_distance(point: (f32, f32)) -> f32{

**// your code here**

}

Inside the function, first destructure the tuple into (x,y). This provides a better readability instead of using point.0 and point.1. Next compute the distance from the original using the formula of √(*x* − *0*)2 + (*y* − *0*)2.  you may consider using the following two functions 1).  x.powf(2.0) for computing the square of the number x. 2). x.sqrt() which will compute the square root of a number x.

Test the program with the following main program

fn main() {

println!("The distance of the point from the origin is {}", print\_distance((5.0,4.0)));

// Note: we need to enclose the inputs to the function in double paranthesis, i.e., print\_distance((5.0,4.0)).

// This is becuase a single paranthesis will mean two inputs of 5.0 and 4.0 and since the function has one

// input which is a single tuple therefore the compiler will complain.  }

1. Complete the following code given below by filling in the code corresponding to the  comments

 fn main(){

    // Step 1: Write a print statement for asking user to input the width of a rectangle

    // Your code here

    // Step 2: Write code for taking the input from the user of type u32 and store it in the variable of width

    // your code here

    // Step 3: Write a print statement for asking the user to input the length of a rectangle

    // Your code here

    // Step 4: Write code for taking the input from the user of type u32 and store it in the variable of length

    // your code here

    let resultant\_area = {

        // Step 5: call a function area() inside here with inputs of width and length which will return the area

    };

    // Step 6: write code to print the resultant\_area variable to the terminal

 }

 fn area(length:u32, width:u32) -> u32 {

    // Step 7: write the definition of the area here which is length \* width and return the result

 }

1. In this exercise question, we will be writting some code that will help us cook an awesome pizza from a given cooking book.

You are required to write four functions corresponding to some tasks. Some of the tasks may be related to each other also.

**Function 1.**

Write a function called expected\_minutes\_in\_oven that will return a value which indicates how many minutes the pizza should be in the oven. According to the cooking book, the expected oven time in minutes is 40: Below is the skeleton of the function indicating what is expected from the function to return.

expected\_minutes\_in\_oven()

// Returns: 40

**Function 2**

Write a function called remaining\_minutes\_in\_oven that takes the actual minutes the pizza has been in the oven as an input  parameter and returns how many minutes the pizza  still has to remain in the oven, based on the expected oven time in minutes from the previous task.

remaining\_minutes\_in\_oven(30)

// Returns: 10 becuase 40- 30 = 10.

**Function 3**

Write a function called preparation\_time\_in\_minutes that takes the number of toppings you added to the pizza as a parameter and returns how many minutes you spent preparing the pizza, assuming each layer takes you 2 minutes to prepare.

preparation\_time\_in\_minutes(2)

// Returns: 4

**Function 4**

Write a function called elapsed\_time\_in\_minutes that takes two input parameters: the first parameter is the number of toppings you added to the pizza, and the second parameter is the number of minutes the pizza has been in the oven. The function should return how many minutes you've worked on cooking the pizza, which is the sum of the preparation time in minutes, and the time in minutes the pizza  has spent in the oven at the moment.

elapsed\_time\_in\_minutes(3, 20)

// Returns: 26

1. Consider the program below. Modify the definition of the quadruple function below by calling the double function twice (this means that hte quadruple function should only make call to the double function and it should call it twice). The quadruple function should return 4 times the number that has been provided to it as an input

fn double(x: i32) -> i32 {

    x \* 2

}

fn quadruple(x: i32)-> i32 {

    // your code here which will call the double function

}

fn main() {

    println!("For 1: the expected value is 4 while the output is {}", quadruple(1));

    println!("For 2: the expected value is 8 while the output is {}", quadruple(2));

    println!("For 3: the expected value is 12 while the output is {}", quadruple(3));

    println!("For 4: the expected value is 16 while the output is {}", quadruple(4));

}