1. Create a struct for a warehouses inventory items. The struct should contain a name, price paid, list price, and quantity.

struct Inventory {

name: String,

total\_price\_paid: f32,

total\_list\_price: f32,

quantity: i32

}

1. Write the implementation block for the following functions: new (to initialize a new instance with certain values), change\_quantity (to increment/decrement its quantity by a given amount) and unit\_profit (to calculate the profit made per unit).

impl Inventory {

fn new(name: String, paid: f32, list: f32, quantity: i32) -> Inventory {

Inventory {

name,

total\_price\_paid: paid,

total\_list\_price: list,

quantity

}

}

fn change\_quantity(&mut self, change: f32) {

self.quantity += change;

}

fn unit\_profit(&self) -> f32 {

return (total\_list\_price – total\_price\_paid) / quantity;

}

}

1. Instantiate an Inventory structure with your new function, change its quantity by -1, and print its unit profit

let mut item = Inventory::new(String::from(“Test”), 10.2, 15.3, 8);

item.change\_quantity(-1)

println!(“Unit Profit: {}”, item.unit\_profit());

1. Explain why the following code will fail to compile, and explain how to fix it

struct Cadet {

name: String,

gpa: f32,

mpa: f32,

pea: f32

}

impl Cadet {

fn new(name: String) -> Cadet {

Cadet { name, 0, 0, 0}

}

fn study(&mut self) {

gpa += 0.5;

}

}

let snoopy = Cadet::new();

snoopy.study();

The study method on Cadet can only be called on a mutable instance of cadet, but snoopy is declared as an immutable variable. This could be fixed by correcting snoopy’s declaration to let mut snoopy = …;