1. Explain why the following code will fail to compile:

fn maybe\_multiply(num: Option<i32>, mult: i32) -> Option<i32> {

let Some(x) = num;

return Some(mult \* x);

}

There is no guarantee that num isn’t None. Even if you have good reason to believe that None will never be passed to the function, Rust will still not allow the program to compile.

1. Correct the code using an if let statement.

fn maybe\_multiply(num: Option<i32>, mult: i32) -> Option<i32> {

if let Some(x) = num {

Some(x \* mult)

}

else {

None

}

}

1. Convert the following match statements to if-lets

match value {

Value::Number(x) => doSomething(x),

\_ => doSomethingElse(),

}

If let Value::Number(x) = value {

doSomething(x);

}

else {

doSomethingElse();

}

match optionalFile {

Some(file) => {

file.open();

file.append(“Hello World\n”);

file.close();

},

None => None

}

If let Some(file) = optionalFile {

file.open();

file.append(“Hello World\n”);

file.close();

}