Errata for *The Rust Programming Language (Covers Rust 2018)* (updated to 5th printing)

Page 3: The URL in the sentence:

"The easiest way to acquire the build tools is to install Build Tools for Visual Studio 2019 at *https://www.visualstudio.com/downloads/#build-tools-for-visual-studio-2019*."

should now read:

https://visualstudio.microsoft.com/visual-cpp-build-tools/

Page 6: The first paragraph:

"At the time of this writing . . . on your computer!"

has been replaced with the following:

"If you want to stick to a standard style across Rust projects, you can use an automatic formatter tool called `rustfmt` to format your code in a particular style. The Rust team has included this tool with the standard Rust distribution, like `rustc`, so it should already be installed on your computer!"

Page 18: In the first code block with the cargo run output:

```
warning: unused 'std::result::Result' which must be used
```

should now read:

```
warning: unused 'Result' which must be used
```

Page 42: The third code block should now read:

```
fn main() {
      let a = [1, 2, 3, 4, 5];
      let first = a[0];
}
```

and the sentence: "The variable named `second` will get the value `2` from index `[1]` in the array" was deleted.

And the first paragraph under "Invalid Array Element Access" was changed to:

“Consider this example that uses code similar to the guessing game in Chapter 2 to ask the user to enter an array index:”

And the last code block on the page should now read:

```
use std::io;

fn main() {
      let a = [1, 2, 3, 4, 5];

      println!("Please enter an array index.");

      let mut index = String::new();
      io::stdin().read_line(&mut index).expect("Failed to read line");
      let index: usize = index.trim().parse().expect(""Not a number");

      let element = a[index];

      println!(
            "The value of the element at index {} is: {}",
            index, element
      );
}
```

Page 43: The first line should now read:

“This code compiles successfully. If you run this code using cargo run and enter 0, 1, 2, 3, or 4, the program will print out the corresponding value at that index in the array. If you instead enter a number past the end of the array, such as 10, you'll see output like this:”

And in the following code block, we changed the content to:

```
thread 'main' panicked at 'index out of bounds: the len is 5 but the index is
10', src/main.rs:12:19
```

And the second paragraph should now read:

“The program resulted in a *runtime* error at the point of using an invalid value in the indexing operation. The program exited with an error message and didn't execute the final `println!`

statement. When you attempt to access an element using indexing, Rust will check that the index you've specified is less than the array length. If the index is greater than or equal to the length, Rust will panic. This check has to happen at runtime, especially in this case, because the compiler can't possibly know what value a user will enter when they run the code later."

Page 78: In Figure 4-6, in the "s" table, the two instances of "5" should now be "11"

Page 156: In the last code block with the error output:

```
expected u32, found enum 'std::result::Result'
```

should now read:

```
expected u32, found enum 'Result'
```

and:

```
found type 'std::result::Result<:fs::file std::io::error="">'
```

should now read:

```
found type 'Result<:fs::file std::io::error="">'
```

Page 205: The sentence in the last paragraph:

"Chapter 19 covers more complex scenarios involving lifetime annotations as well as some advanced type system features"

should now read:

"There are also more complex scenarios involving lifetime annotations that you will only need in very advanced scenarios; for those, see the reference at *https://doc.rust-lang.org/stable/reference/trait-bounds.html#lifetime-bounds*"

Page 248: In the first code block with the warning:

```
warning: unused 'std::result::Result' that must be used
```

should now read:

```
warning: unused 'Result' that must be used
```

Pages 398-399: The line:

"The first call to enumerate produces the type `(0, 'a')`."

should now read:

“The first value produced is the tuple `(0, 'a')`.”

Page 411: In the last sentence of the first paragraph, “but when we run this code” should now read “but when we compile this code”

Page 459: The line:
“We’ve chosen this port for two reasons: HTTP is normally accepted on this port, and 7878 is rust typed on a telephone.”
should now read:
“We’ve chosen this port for two reasons: HTTP isn’t normally accepted on this port, and 7878 is rust typed on a telephone.”

And “1024” should now read “1023”

Page 461-463: In Listing 20-2 and in the third paragraph on page 461, and in Listing 20-3 on page 463, “512” should be changed to “1024”

Page 464: Listing 20-5 should now read:

```
use std::fs;
// --snip--
fn handle_connection(mut stream: TcpStream) {
      let mut buffer = [0; 1024];
      stream.read(&mut buffer).unwrap();

      let contents = fs::read_to_string("hello.html").unwrap();
      let response = format!(
            "HTTP/1.1 200 OK\r\nContent-Length: {}\r\n\r\n{}",
            contents.len(), contents
      );
      stream.write(response.as_bytes()).unwrap();
      // --snip--
```

Page 465: The first sentence should now read:

“Next, we use `format!` to include a Content-Length header and the file’s contents in the response.”

And Listing 20-6 should now read:

```
// --snip--

fn handle_connection(mut stream: TcpStream) {
      let mut buffer = [0; 1024];
      stream.read(&mut buffer).unwrap();

      let get = b"GET / HTTP/1.1\r\n";

      if buffer.starts_with(get) {
            let contents = fs::read_to_string(«hello.html»).unwrap();
            // --snip--
            stream.flush().unwrap();
      } else {
            // some other request
      }
}
```

Page 466: Listing 20-7 should now read:

```
// --snip--

} else {
      let status_line = "HTTP/1.1 404 NOT FOUND";
      let contents = fs::read_to_string("404.html").unwrap();
      let response = format!(
            "{}\r\nContent-Length: {}\r\n\r\n{}",
            status_line, contents.len(), contents
      );
      stream.write(response.as_bytes()).unwrap();
      stream.flush().unwrap();
}
```

And the Listing 20-7 caption should now read:

“Responding to any other request with status code 404 and an error page”

And the line:
“We’re still not returning headers, and the body of the response will be the HTML in the file *404.html*.”
should now read:
“The body of the response will be the HTML in the file *404.html*.”

Page 467: In Listing 20-9, we deleted the two instances of: `\r\n\r\n`

And we replaced the tenth code line, `let response`, with:

```
let response = format!(
      "{}\r\nContent-Length: {}\r\n\r\n{}",
      status_line, contents.len(), contents
);
```

And the Listing 20-9 caption should now read:
“Refactoring the `if` and `else` blocks to contain only the code that differs”

Page 468: In Listing 20-10, we deleted the three instances of `\r\n\r\n`