**Interpretation challenge**

The following sample Rust program has been written to parse a str slice as an integer and prove this has worked by incrementing it.

Of course, if the parse fails, a run-time error will occur in the form of a Rust panic! So, it must be correctly error-handled via propagation.

The following code has been submitted by a member of your team as PoC and you have been asked to review it and answer some questions.



fn parse\_as\_int(num: &str) -> Option<i32> {

Some(num.parse::<i32>().ok()?)

}

fn main() {

let result = parse\_as\_int("42");

match result {

Some(int\_val) => println!("Number +1 is {}", int\_val+1),

None => println!("Not a number!"),

}

}

Try to answer these *without* executing the code first. Once answered, feel free to compare answers with a peer or run the code to see if you are correct.

**Questions:**

1. What code is being replaced by line 2’s use of the try (?) operator?
2. Why is the .ok() method required here?
3. What happens if we pass a floating-point number to the function instead?
4. What happens if an empty string slice is passed to the function instead?
5. What happens if the number 2147483647 is passed to the function instead? Why? (Be careful with this one…)