CS 551 Spring 2025 Quiz 3 KEY

February 19, 2025

Name:

Email:

Instructions: Put your name and email in the appropriate places. Answer, to the best of your ability the question(s) below. Additionally, sign your name below the academic honesty statement. If you deviate from these instructions in any way, you will receive a zero on the quiz.

Unless otherwise specified, you can assume that all necessary imports have been made and there are no deliberate typos in function or type names.

Additional information: An `Option<T>` is an enum that has two variants: `Some(T)` and None, where `T` is a type known at compile time.

Consider the following stucts, traits, and enums:

```
1  pub trait IntoIterator {
2      /// Creates an 'Iterator' from a value.
3      fn into_iter(self) -> Self::IntoIter;
4  }
5
6  pub trait Iterator {
7      /// Advances the iterator and returns the next value.
8      fn next(&mut self) -> Option<Self::Item>;
9  }
10
11 pub enum FizzBuzz<T> {
12     Fizz,
13     Buzz,
14     Both,
15     Neither(T),
16 }
```

Will the following program compile:

```
1 fn main() {
2     let args = std::env::args().collect::<Vec<_>>();
3     let mut args_iter = args.into_iter();
4     let arg = args_iter.next();
5
6     let arg = args_iter.next();
7 }
```

(circle your answer)

Yes

Reason: although `into_iter(self)` takes ownership of `args` on line 3, `next(&self)` only borrows `self`, and thus it's fine to call it multiple times.

Will the following program compile:

```
1 fn main() {
2     let args = std::env::args().collect::<Vec<_>>();
3     let mut args_iter = args.into_iter();
4     let arg = args_iter.next();
5
6     let args_iter = args.into_iter();
7 }
```

(circle your answer)

No

Reason: `into_iter(self)` takes ownership of `args` on line 3 and never gives it back Thus the second call to `args.into_iter()` (line 6) is a use after move.

Will the following program compile:

```
1 fn main() {
2     let fizz = FizzBuzz::<i32>::Fizz;
3     match fizz {
4         FizzBuzz::Fizz => println!("fizz"),
5         FizzBuzz:Buzz => println!("buzz"),
6         _ => println!("???"),
7     }
8 }
```

(circle your answer)

Yes

Reason: Although every variant of `FizzBuzz` is not explicitly handled in a match arm, line 6 uses the wild card pattern (_) to handle any remaining variants.

Will the following program compile:

```
1 fn main() {
```

```
2       let fizz = FizzBuzz::<i32>::Fizz;
3       match fizz {
4           FizzBuzz::Fizz => println!("fizz"),
5           FizzBuzz:Buzz => println!("buzz"),
6           FizzBuzz::Both => println!("???"),
7       }
8   }
```

(circle your answer)

No

Reason: The `FizzBuzz::Neither(T)` variant is not handled in the `match` statement.

Academic honesty statement: I have done this quiz completely on my own. I have not copied it from, nor have I given answers to anyone else. I understand that if I am involved in plagiarism or cheating I will have to sign an official form that I have cheated and that this form will be stored in my official university record. I also understand that I will receive a grade of 0 for the quiz involved, my grade in the class will be reduced by at least one level (e.g., from A to B) for my offense, and that I will receive a grade of "F" for the course for any additional offense of any kind.