**PC-1**

1 #![allow(unused\_variables)]

2

3 #[derive(Debug)]

4 struct Inner {

5 in\_a: u8,

6 in\_b: u8

7 }

8

9 struct Outer1 {

10 a: [Inner; 2]

11 }

12

13 struct Outer2 {

14 a: (Inner, Inner)

15 }

16

17 fn test(num: &mut u8, inner: &Inner) {

18 \*num += 1;

19 println!("{:?}", inner);

20 }

21

22 fn main() {

23

24 let mut out1 = Outer1 {

25 a: [Inner {in\_a: 1, in\_b: 2}, Inner {in\_a: 3, in\_b: 4}]

26 };

27

28 let mut out2 = Outer2 {

29 a: (Inner {in\_a: 1, in\_b: 2}, Inner {in\_a: 3, in\_b: 4})

30 };

31

32 test(&mut out1.a[0].in\_a, &out1.a[1]);

33 test(&mut out2.a.0.in\_a, &out2.a.1);

34 }

35

Original compile error:

**error[E0502]: cannot borrow `out1.a[\_]` as immutable because it is also borrowed as mutable**

**-->** main.rs:32:31

**|**

**32** **|**  test(&mut out1.a[0].in\_a, &out1.a[1]);

**|**  **----** **-------------------** **^^^^^^^^^^** **immutable borrow occurs here**

**|**  **|** **|**

**|**  **|** **mutable borrow occurs here**

**|**  **mutable borrow later used by call**

**error: aborting due to previous error**

**For more information about this error, try `rustc --explain E0502`.**

Extended compile error:

**error[E0502]: cannot borrow `out1.a[\_]` as immutable because it is also borrowed as mutable**

**-->** main.rs:32:31

**|**

**32** **|**  test(&mut out1.a[0].in\_a, &out1.a[1]);

**|**  **----** **-------------------** **^^^^^^^^^^** **immutable borrow occurs here**

**|**  **|** **|**

**|**  **|** **mutable borrow occurs here**

**|**  **mutable borrow later used by call**

**note**: `a` is an array and can only be borrowed as a whole

**|**

**9 |** struct Outer1 {

**10 |** a: [Inner; 2]

**|** **^**

**error: aborting due to previous error**

**For more information about this error, try `rustc --explain E0502`.**

Q6 grading rubric:

| **Attribute** | **Poor** | **Ok** | **Good** |
| --- | --- | --- | --- |
| What Rust safety Concept | Does not know it is a problem about borrow | Mentioned borrow, but does not distinguish mutability | Mentioning “cannot have more than one borrow when there’s one mutable borrow” |
| How Rust Concept Use in Specific Program Construct | Does not know the problem is because of array | Know there are some problems with array, but does not know exactly how it couldn’t be borrowed | Mention how array couldn’t be borrowed as a whole, or structs can be partially borrowed |

**PC-2**

1 #![allow(unused\_variables)]

2

3 #[derive(Debug)]

4 struct Inner {

5 in\_a: u8,

6 in\_b: u8

7 }

8

9 struct Outer1 {

10 a: [Inner; 2]

11 }

12

13 struct Outer2 {

14 a: (Inner, Inner)

15 }

16

17 fn test(inner: Inner, num: &mut u8) {

18 \*num += 1;

19 println!("{:?}", inner);

20 }

21

22 fn main() {

23 let mut out1 = Outer1 {

24 a: [Inner {in\_a: 1, in\_b: 2}, Inner {in\_a: 3, in\_b: 4}]

25 };

26 let mut out2 = Outer2 {

27 a: (Inner {in\_a: 1, in\_b: 2}, Inner {in\_a: 3, in\_b: 4})

28 };

29

30 test(out1.a[1], &mut out1.a[0].in\_a);

31 test(out2.a.1, &mut out2.a.0.in\_a);

32 }

Compile error:

**error[E0508]: cannot move out of type `[Inner; 2]`, a non-copy array**

**-->** main.rs:30:10

**|**

**30** **|**  test(out1.a[1], &mut out1.a[0].in\_a);

**|**  **^^^^^^^^^**

**|**  **|**

**|**  **cannot move out of here**

**|**  **move occurs because `out1.a[\_]` has type `Inner`, which does not implement the `Copy` trait**

**error: aborting due to previous error**

**For more information about this error, try `rustc --explain E0508`.**

Q6 grading rubric:

| **Attribute** | **Poor** | **Ok** | **Good** |
| --- | --- | --- | --- |
| What Rust safety Concept | Does not know it is a problem about move semantics | Mentioned one of copy or move, but unclear | Explicitly mention it is move vs copy and give a correct explanation |
| How Rust Concept Use in Specific Program Construct | Does not know the problem is because of array | Know there are some problems with array, but does not know exactly how it couldn’t be moved | Mention how array is different from other data structures, or explicitly explain the root cause and why cannot move an element from array |

**PC-3**

1 #![allow(unused\_variables)]

2

3 #[derive(Debug)]

4 struct Inner {

5 in\_a: u8,

6 in\_b: u8

7 }

8

9 struct Outer1 {

10 a: [Inner; 2]

11 }

12

13 struct Outer2 {

14 a: (Inner, Inner)

15 }

16

17 fn main() {

18 let mut out1 = Outer1 {

19 a: [Inner {in\_a: 1, in\_b: 2}, Inner {in\_a: 3, in\_b: 4}]

20 };

21 let mut out2 = Outer2 {

22 a: (Inner {in\_a: 1, in\_b: 2}, Inner {in\_a: 3, in\_b: 4})

23 };

24 let r1 = &mut out1.a[0].in\_a;

25 let r3 = &mut out2.a.0.in\_a;

26 let r2 = &out1.a[1];

27 let r4 = &out2.a.1;

28 \*r1 += 1;

29 \*r3 += 1;

30

31 println!("{:?}", r2);

32 println!("{:?}", r4);

33 }

Compile error:

**error[E0502]: cannot borrow `out1.a[\_]` as immutable because it is also borrowed as mutable**

**-->** main.rs:26:14

**|**

**24** **|**  let r1 = &mut out1.a[0].in\_a;

**|**  **-------------------** **mutable borrow occurs here**

**25** **|**  let r3 = &mut out2.a.0.in\_a;

**26** **|**  let r2 = &out1.a[1];

**|**  **^^^^^^^^^^** **immutable borrow occurs here**

**27** **|**  let r4 = &out2.a.1;

**28** **|**  \*r1 += 1;

**|**  **--------** **mutable borrow later used here**

**error: aborting due to previous error**

**For more information about this error, try `rustc --explain E0502`.**

Q6 grading rubric:

| **Attribute** | **Poor** | **Ok** | **Good** |
| --- | --- | --- | --- |
| What Rust safety Concept | Does not know it is a problem about borrow | Mentioned borrow, but does not distinguish mutability | Mentioning “cannot have more than one borrow when there’s one mutable borrow” |
| How Rust Concept Use in Specific Program Construct | Does not know the problem is because of array | Know there are some problems with array, but does not know exactly how it couldn’t be borrowed | Mention how array couldn’t be borrowed as a whole |

**PD-1**

1 #![allow(unused\_variables)]

2

3 mod case2 {

4 #[derive(Debug)]

5 struct Foo {}

6

7 struct Bar2<'b> {

8 x: &'b Foo,

9 i: i32

10 }

11

12 impl<'b> Bar2<'b> {

13 fn f1(&'b mut self) -> &'b Foo {

14 println!("{:?}", self.x);

15 self.x

16 }

17

18 fn f2<'a>(&self, s: &'a str, ind: usize) -> &'a str {

19 println!("{:?}, {}", self.x, s);

20 &s[..ind]

21 }

22

23 }

24

25 fn f4() {

26 let foo = Foo {};

27 let mut bar2 = Bar2 { x: &foo, i: 1 };

28 let excerpt = bar2.f2("An online article", 2);

29 println!("{}", excerpt);

30 bar2.f1();

31 let z = bar2.f1();

32 }

33 }

34

35 fn main() {}

Original compile error:

**error[E0499]: cannot borrow `bar2` as mutable more than once at a time**

**-->** main.rs:31:17

**|**

**30** **|**  bar2.f1();

**|**  **----** **first mutable borrow occurs here**

**31** **|**  let z = bar2.f1();

**|**  **^^^^**

**|**  **|**

**|**  **second mutable borrow occurs here**

**|**  **first borrow later used here**

**error: aborting due to previous error**

**For more information about this error, try `rustc --explain E0499`.**

Extended compile error:

**error[E0499]: cannot borrow `bar2` as mutable more than once at a time**

**-->** main.rs:31:17

**|**

**30** **|**  bar2.f1();

**|**  **----** **first mutable borrow occurs here**

**31** **|**  let z = bar2.f1();

**|**  **^^^^**

**|**  **|**

**|**  **second mutable borrow occurs here**

**|**  **first borrow later used here**

**note**: the lifetime of reference `self` is the same as struct `Bar2`

**|**

**12 |** impl<'b> Bar2<'b> {

**|** **-- note**: `'b` is the lifetime parameter of struct `Bar2`

**13 |** fn f1(&'b mut self) -> &'b Foo {

**|** **^^**

**|** **|**

**|** **note**: ...so this lifetime parameter makes the lifetime of

reference `self` the same as struct `Bar2`

**error: aborting due to previous error**

**For more information about this error, try `rustc --explain E0499`.**

Q6 grading rubric:

| **Attribute** | **Poor** | **Ok** | **Good** |
| --- | --- | --- | --- |
| What Rust safety Concept | Doesn’t mention about borrowing or reference | Mentioned problem with mutable borrowing | Mention mutable borrowing for more than once |
| How Rust Concept Use in Specific Program Construct | Does not know it is a problem about the receiver | Mention `&mut self`’s lifetime could be problematic, but doesn’t know more; or understand where lifetime ends but explanation incorrect | Explain that in f2, &self has the same lifetime parameter as struct Bar2, so the borrow to self lasts as long as the struct itself |

**PD-2**

1 #![allow(unused\_variables)]

2

3 mod case2 {

4 #[derive(Debug)]

5 struct Foo {}

6

7 struct Bar2<'b> {

8 x: &'b Foo,

9 i: i32

10 }

11

12 impl<'b> Bar2<'b> {

13 fn f1(self) -> i32 {

14 println!("{:?}", self.x);

15 self.i

16 }

17

18 fn f2<'a>(&self, s: &'a str, ind: usize) -> &'a str {

19 println!("{:?}, {}", self.x, s);

20 &s[..ind]

21 }

22

23 }

24

25 fn f4() {

26 let foo = Foo {};

27 let bar2 = Bar2 { x: &foo, i: 1 };

28 let excerpt = bar2.f2("An online article", 2);

29 println!("{}", excerpt);

30 bar2.f1();

31 let z = bar2.f1();

32 }

33 }

34

35 fn main() {}

Compile error:

**error[E0382]: use of moved value: `bar2`**

**-->** main.rs:31:17

**|**

**27** **|**  let bar2 = Bar2 { x: &foo, i: 1 };

**|**  **----** **move occurs because `bar2` has type `Bar2<'\_>`, which does not implement the `Copy` trait**

**...**

**30** **|**  bar2.f1();

**|**  **----** **`bar2` moved due to this method call**

**31** **|**  let z = bar2.f1();

**|**  **^^^^** **value used here after move**

**|**

**note**: this function consumes the receiver `self` by taking ownership of it, which moves `bar2`

**-->** main.rs:13:15

**|**

**13** **|**  fn f1(self) -> i32 {

**|**  **^^^^**

**error: aborting due to previous error**

**For more information about this error, try `rustc --explain E0382`.**

Q6 grading rubric:

| **Attribute** | **Poor** | **Ok** | **Good** |
| --- | --- | --- | --- |
| What Rust safety Concept | Doesn’t mention about move self | Mention move self but no more | Besides move, also mentioned Copy could solve the problem |
| How Rust Concept Use in Specific Program Construct | Does not know it is a problem about function + struct | Mention moving `self` could be problematic, but doesn’t know more | Explicitly explain why move self is incorrect (e.g. potential use after move, you cannot use it after move) |

**PD-3**

1 #![allow(unused\_variables)]

2

3 mod case2 {

4 #[derive(Debug)]

5 struct Foo {}

6

7 struct Bar2<'b> {

8 x: &'b Foo,

9 i: i32

10 }

11

12 fn f4() {

13 let foo = Foo {};

14 let mut bar2 = Bar2 { x: &foo, i: 1 };

15 let f2: for<'a> fn(&Bar2, &'a str, usize) -> &'a str =

16 |bar: &Bar2, s: &str, ind: usize| {

17 println!("{:?}, {}", bar.x, s);

18 &s[..ind]

19 };

20 let f3: for<'a> fn(&'a mut Bar2<'a>) -> &'a Foo =

21 |bar: &mut Bar2| {

22 println!("{:?}", bar.x);

23 bar.x

24 };

25 let excerpt = f2(&bar2, "An online article", 2);

26 println!("{}", excerpt);

27 f3(&mut bar2);

28 let z = f3(&mut bar2);

29 }

30 }

31

32 fn main() {}

Compile error:

**error[E0499]: cannot borrow `bar2` as mutable more than once at a time**

**-->** main.rs:28:20

**|**

**27** **|**  f3(&mut bar2);

**|**  **---------** **first mutable borrow occurs here**

**28** **|**  let z = f3(&mut bar2);

**|**  **^^^^^^^^^**

**|**  **|**

**|**  **second mutable borrow occurs here**

**|**  **first borrow later used here**

**error: aborting due to previous error**

**For more information about this error, try `rustc --explain E0499`.**

Q6 grading rubric:

| **Attribute** | **Poor** | **Ok** | **Good** |
| --- | --- | --- | --- |
| What Rust safety Concept | Doesn’t mention about borrowing or reference | Mentioned problem with mutable borrowing | Mention mutable borrowing for more than once |
| How Rust Concept Use in Specific Program Construct | Does not know it is a problem about closure/function | Knowing it is about misuse of lifetime annotation in function pointer types | Explain that “In type annotation of f3, the lifetime parameter of the argument reference is the same as the struct Bar2” |