# Rust Quiz

- 4 questions
- 50 seconds per question

**David Tolnay & Alex Crichton**

1

```rust
struct S {
    f: fn(),
}

impl S {
    fn f(&self) {
        print!("1");
    }
}

fn main() {
    let print2 = || print!("2");
    S { f: print2 }.f();
}
```

1

1

```
struct S {
    f: fn(),
}

impl S {
    fn f(&self) {
        print!("1");
    }
}

fn main() {
    let print2 = || print!("2");
    S { f: print2 }.f();
}
```

1

1

```
struct S {
    f: fn(),
}

impl S {
    fn f(&self) {
        print!("1");
    }
}

fn main() {
    let print2 = || print!("2");
    S { f: print2 }.f();
    (S { f: print2 }.f)();
}
```

2

2

```
struct D(u8);

impl Drop for D {
    fn drop(&mut self) {
        print!("{}", self.0);
    }
}

struct S {
    d: D,
    x: u8,
}

fn main() {
    let S { x, .. } = S { d: D(1), x: 2 };
    print!("{}", x);
    let S { ref x, .. } = S { d: D(3), x: 4 };
    print!("{}", x);
}
```

```rust
struct D(u8);

impl Drop for D {
    fn drop(&mut self) {
        print!("{}", self.0);
    }
}

struct S {
    d: D,
    x: u8,
}

fn main() {
    let S { x, .. } = S { d: D(1), x: 2 };
    print!("{}", x);
    let S { ref x, .. } = S { d: D(3), x: 4 };
    print!("{}", x);
}
```

2

1243

3

```
struct S(i32);

impl std::ops::BitAnd<S> for () {
    type Output = ();

    fn bitand(self, rhs: S) {
        print!("{}", rhs.0);
    }
}

fn main() {
    let f = || ( () & S(1) );
    let g = || { () & S(2) };
    let h = || ( {} & S(3) );
    let i = || { {} & S(4) };
    f();
    g();
    h();
    i();
}
```

3

123

```
struct S(i32);

impl std::ops::BitAnd<S> for () {
    type Output = ();

    fn bitand(self, rhs: S) {
        print!("{}", rhs.0);
    }
}

fn main() {
    let f = || ( () & S(1) );
    let g = || { () & S(2) };
    let h = || ( {} & S(3) );
    let i = || { {} & S(4) };
    f();
    g();
    h();
    i();
}
```

3

123

```
struct S(i32);

impl std::ops::BitAnd<S> for () {
    type Output = ();

    fn bitand(self, rhs: S) {
        print!("{}", rhs.0);
    }
}

fn main() {
    let f = || ( () & S(1) );
    let g = || { () & S(2) };
    let h = || ( {} & S(3) );
    let i = || {
        {}
        &S(4)
    };
    f();
    g();
```

```
#[repr(u8)]
enum Enum {
    First = 0,
    Second = 1,
}

impl Enum {
    fn p(self) {
        match self {
            First => print!("1"),
            Second => print!("2"),
        }
    }
}

fn main() {
    Enum::p(unsafe { std::mem::transmute(1u8) });
}
```

1

```
#[repr(u8)]
enum Enum {
    First = 0,
    Second = 1,
}

impl Enum {
    fn p(self) {
        match self {
            First => print!("1"),
            Second => print!("2"),
        }
    }
}

fn main() {
    Enum::p(unsafe { std::mem::transmute(1u8) });
}
```

4

1

```
warning: unreachable pattern
  --> src/main.rs:12:13
   |
11 |             First => print!("1"),
   |             ----- matches any value
12 |             Second => print!("2"),
   |             ^^^^^^ unreachable pattern
   |
   = note: #[warn(unreachable_patterns)] on by default
```

```
macro_rules! m {
    ($($t:tt)*) => {
        stringify!($($t)*=*)
    }
}

fn main() {
    println!("{}", m!(a b));
}
```

```
macro_rules! m {
    ($($t:tt)*) => {
        stringify!($($t)*=*)
    }
}

fn main() {
    println!("{}", m!(a b));
}
```

a *= b

a *= b

```
macro_rules! m {
    ($($t:tt)*) => {
        stringify!($($t) *= *)
    }
}

fn main() {
    println!("{}", m!(a b));
}
```

6

```
pub trait Trait {
    fn f(&self);
}

impl<'a> dyn Trait + 'a {
    pub fn f(&self) {
        print!("1");
    }
}

impl Trait for bool {
    fn f(&self) {
        print!("2");
    }
}

fn main() {
    Trait::f(&true);
    Trait::f(&true as &dyn Trait);
    <_ as Trait>::f(&true);
    <_ as Trait>::f(&true as &dyn Trait);
    <bool as Trait>::f(&true);
}
```

```
pub trait Trait {
    fn f(&self);
}

impl<'a> dyn Trait + 'a {
    pub fn f(&self) {
        print!("1");
    }
}

impl Trait for bool {
    fn f(&self) {
        print!("2");
    }
}

fn main() {
    Trait::f(&true);
    Trait::f(&true as &dyn Trait);
    <_ as Trait>::f(&true);
    <_ as Trait>::f(&true as &dyn Trait);
    <bool as Trait>::f(&true);
}
```

22222

```
struct S {
    x: i32,
}

const S: S = S { x: 2 };

fn main() {
    let v = &mut S;
    v.x += 1;
    S.x += 1;
    print!("{}{}", v.x, S.x);
}
```

7

32

```
struct S {
    x: i32,
}

const S: S = S { x: 2 };

fn main() {
    let v = &mut S;
    v.x += 1;
    S.x += 1;
    print!("{}{}", v.x, S.x);
}
```