Note: this error code is no longer emitted by the compiler. In a pattern, all values that don't implement the `Copy` trait have to be bound the same way. The goal here is to avoid binding simultaneously by-move and by-ref.

This limitation may be removed in a future version of Rust.

Erroneous code example:

```
#![feature(move_ref_pattern)]

struct X { x: (), }

let x = Some((X { x: () }, X { x: () }));
match x {
    Some((y, ref z)) => {}, // error: cannot bind by-move and by-ref in the
                            //        same pattern
    None => panic!()
}
```

You have two solutions:

Solution #1: Bind the pattern's values the same way.

```
struct X { x: (), }

let x = Some((X { x: () }, X { x: () }));
match x {
    Some((ref y, ref z)) => {},
    // or Some((y, z)) => {}
    None => panic!()
}
```

Solution #2: Implement the `Copy` trait for the `X` structure.

However, please keep in mind that the first solution should be preferred.

```
#[derive(Clone, Copy)]
struct X { x: (), }

let x = Some((X { x: () }, X { x: () }));
match x {
    Some((y, ref z)) => {},
    None => panic!()
}
```