```
[[questions]]
type = "Tracing"
prompt.program = """
fn main() {
    let x = null;
    foo(x);
}
fn foo(x:u8) {
    println!("{}", x);
}
"""
answer.doesCompile = false
context = """
Cairo does not have null pointers, so the `null` keyword does not exist.
An `Option` type should be used to represent the possibility of an object being null.
"""
id = "8b786183-ef9f-43f8-89a6-0e9c0e87c576"
[[questions]]
type = "MultipleChoice"
prompt.prompt = """
Consider these two representations of a `Result` type that contains a value `T` if a
computation succeeds, or an error `E` if it fails.
```
struct Result1<T, E> {
    ok: Option<T>,
    err: Option<E>,
}
enum Result2<T, E> {
    Ok : T,
    Err : E,
}
```
The enum `Result2` is considered more idiomatic than the struct `Result1` in Cairo.
Which statement below is **NOT** a valid reason why?
"""
prompt.distractors = [
  "The struct is more syntactically verbose to construct than the enum",
  "The struct uses more space in memory at runtime than the enum",
  "The struct could have `ok` and `err` both be `None`, while the enum must have at
least one of them",
]
answer.answer = "The struct contains `Option` types, which are only intended to wrap
structs"
context = """
It's perfectly fine to have structs contain `Option` types as fields (the question asked for
a statement which does **NOT** describe a valid reason). But if your data structure has
```

invariants like "exactly one of two optional fields should be `Some`", then that invariant is better ensured by the use of an enum.
"""

id = "0d9b6f65-bfac-447f-a2d4-a650abc8bc01"