```
[[questions]]
type = "MultipleChoice"
prompt.prompt = "Which one of the following is **NOT** true about smart pointers?"
prompt.distractors = [
  "Smart pointers help prevent common memory issues such as dereferences unallocated memory and re-writing to existing cells.",
  "Smart pointers ensure proper initialization and prevent access to uninitialized or invalid memory.",
  "Smart pointers such as `Box` and `Nullable` enable efficient passing of large data by moving or sharing ownership without expensive copying.",
]
answer.answer = "Smart pointers store a reference to a value but do not provide any automatic memory management or ownership tracking."
context = """
Smart pointers in Cairo possess additional metadata and capabilities beyond merely serving as a reference to a value.
They offer memory management features that extend beyond simple referencing, including strict type checking and ownership rules that enforce memory safety.
Cairo provides several explicit smart pointer types, such as `Box` and `Nullable`, but other types like `Array` of `Felt252Dict` are also a form of smart pointers.
These smart pointers ensure memory safety through strict ownership rules, preventing common issues like null dereferences.
"""
id = "dbf869eb-b27a-41d0-b428-20d9ae00498b"
[[questions]]
type = "Tracing"
prompt.program = """
#[derive(Drop)]
struct Student {
    name: ByteArray,
    age: u8,
    id: u32
}
fn main() {
    let mut student1 = BoxTrait::new(Student { name: "Peter", age: 12, id: 12345 });
    let student2 = student1;
    student1 = BoxTrait::new(Student { name: "James", age: 18, id: 56789 });
    println!("{}", student2.unbox().name);
}
"""
answer.doesCompile = true
answer.stdout = "Peter"
context = """
The `student1` variable was first instantiated as a smart pointer to an instance of the struct `Student`.
When we assigned `student1` to a new variable `student2`, we merely copied the
```

smart pointer to the previously created struct,
so both variables referred to the same struct in memory.
Once `student1` was reinstantiated with a new smart pointer to a new `Student`
instance, the `student2` variable
still referred to the original struct, so printing `student2.name` displayed `"Peter"`.
"""
id = "8d7ac906-4579-4d47-8345-4ec30a5f41f3"
[[questions]]
type = "MultipleChoice"
prompt.prompt = """
Which of the following statement is TRUE when the following program is run with `scarb cairo-run`?
```
use core::nullable::{NullableTrait, match_nullable, FromNullableResult};
fn main() {
    let mut scoreSheet: Felt252Dict<Nullable<Span<felt252>>> = Default::default();
    let exams = array![60, 70, 80, 90];
    scoreSheet.insert(0, NullableTrait::new(exams.span()));
    let firstSession = scoreSheet.get(0);
    let record = match match_nullable(firstSession) {
        FromNullableResult::Null => panic!("No value found"),
        FromNullableResult::NotNull(firstSession) => firstSession.unbox(),
    };
    println!("Exams {}", *record.at(4));
}
```
"""
prompt.distractors = [
  "Outputs `Exams 90`",
  "Outputs 'No value found' panic because it implements the `zero_default` method",
  "Fails to compile",
]
answer.answer = "Panics with an 'Index out of bounds' error"
context = """
The `Index out of bounds` error occurs due to an attempt to access
the fifth element of a four-element array, which is an unallocated memory cell.
"""
id = "df5c2298-4896-459e-ac4a-bc5e47582a13"