```python
# pydantic_type_to_yaml_schema

from agentparse import pydantic_type_to_yaml_schema


# Test mapping of basic Pydantic types to YAML schema types
def test_basic_type_mapping():
    assert pydantic_type_to_yaml_schema(int) == "integer"
    assert pydantic_type_to_yaml_schema(float) == "number"
    assert pydantic_type_to_yaml_schema(str) == "string"
    assert pydantic_type_to_yaml_schema(bool) == "boolean"
    assert pydantic_type_to_yaml_schema(list) == "array"
    assert pydantic_type_to_yaml_schema(dict) == "object"


# Test mapping of a complex type (e.g., Optional)
def test_optional_type_mapping():
    from typing import Optional

    assert pydantic_type_to_yaml_schema(Optional[int]) == "integer"


# Test mapping of a generic type (e.g., List)
def test_generic_list_type_mapping():
    from typing import List
    assert pydantic_type_to_yaml_schema(List[int]) == "array"



# Test mapping of a generic type (e.g., Dict)
def test_generic_dict_type_mapping():
    from typing import Dict

    assert pydantic_type_to_yaml_schema(Dict[str, int]) == "object"



# Test mapping of an unsupported type
def test_unsupported_type_mapping():
    class CustomType:
        pass

    assert pydantic_type_to_yaml_schema(CustomType) == "string"



# Test mapping of a nested structure
def test_nested_structure_mapping():
    from typing import List, Dict

    assert (
        pydantic_type_to_yaml_schema(Dict[str, List[int]]) == "object"
    )
```