task 1:

```
import time
import pytest


@pytest.fixture(scope="session", autouse=True)
def track_suite_time():
    start = time.time()
    print("\n[SUITE] Test suite started...")
    yield
    end = time.time()
    duration = end - start
    print(f"[SUITE] Test suite finished in {duration:.2f} seconds.")


@pytest.fixture()
def track_test_time(request):
    start = time.time()
    print(f"\n[TEST] Test '{request.node.name}' started...")
    yield
    end = time.time()
    duration = end - start
    print(f"[TEST] Test '{request.node.name}' finished in {duration:.2f}
seconds.")


def add_numbers(a, b):
    return a + b


@pytest.mark.usefixtures("track_test_time")
def test_add_two_positive_numbers():
    a, b = 3, 5
    result = add_numbers(a, b)
    time.sleep(2)
    assert result == 8


@pytest.mark.usefixtures("track_test_time")
def test_add_two_negative_numbers():
    a, b = -3, -5
    result = add_numbers(a, b)
    time.sleep(3)
```

```
    assert result == -8


def test_add_negative_and_positive_numbers():
    a, b = -3, 5
    result = add_numbers(a, b)
    time.sleep(10)
    assert result == 2
```

```
============================= test session starts ==============================
collecting ... collected 3 items

task 3.py::test_add_two_positive_numbers
[SUITE] Test suite started...

[TEST] Test 'test_add_two_positive_numbers' started...
PASSED                        [ 33%][TEST] Test 'test_add_two_positive_numbers' finished in 2.00 seconds.

task 3.py::test_add_two_negative_numbers
[TEST] Test 'test_add_two_negative_numbers' started...
PASSED                        [ 66%][TEST] Test 'test_add_two_negative_numbers' finished in 3.00 seconds.

task 3.py::test_add_negative_and_positive_numbers PASSED                [100%][SUITE] Test suite finished in 15.


============================== 3 passed in 15.15s ==============================

Process finished with exit code 0
```

task 2:

```
import pytest
import yaml


# Load test data from YAML file
with open("config.yaml") as f:
    data = yaml.safe_load(f)


def add_numbers(*args):
    return sum(args)


@pytest.mark.smoke
@pytest.mark.parametrize(
    "a,b,c,expected",
    [(case["input"][0], case["input"][1], case["input"][2], case["expected"])
```

```python
    for case in data["cases"]],
        ids=[case["case_name"] for case in data["cases"]]
    )
    def test_add_numbers(a, b, c, expected):
        result = add_numbers(a, b, c)
        assert result == expected


    @pytest.mark.critical
    def test_add_invalid_types():
        with pytest.raises(TypeError):
            add_numbers(1, "two", 3)
```

```
============================= test session starts ==============================
collecting ... collected 4 items

parametrize_task.py::test_add_numbers[positive]
parametrize_task.py::test_add_numbers[negative]
parametrize_task.py::test_add_numbers[mixed]
parametrize_task.py::test_add_invalid_types PASSED                  [ 25%]PASSED                  [ 50%]PASSE

======================== 4 passed, 2 warnings in 1.40s =========================

Process finished with exit code 0
```