ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Факультет информационных технологий
Кафедра Информатики и информационных технологий

направление подготовки
09.03.02 «Информационные системы и технологии»

ЛАБОРАТОРНАЯ РАБОТА № 6

Дисциплина: Тестирование программного обеспечения

Тема: Написание модульных тестов (Unit Tests)

Выполнил(а): студент(ка) группы: 221-3711

До Дык Зунг

(Фамилия И.О.)

Дата, подпись_____ _____
(Дата) (Подпись)

Проверил:_____ _____
(Фамилия И.О., степень, звание) (Оценка)

Дата, подпись_____ _____
(Дата) (Подпись)

Замечания: _____

_____

_____

Москва

2025

Задание (Автоматическое тестирование):

1. **Используя фреймворк `unittest` или `pytest`, напишите модульные тесты для всех методов.**
2. **Используйте технику параметризации для тестирования с разными наборами данных.**
3. **Для метода `divide` убедитесь, что тест проверяет возникновение исключения (exception) при делении на ноль.**

calculator.py

```
from __future__ import annotations

class Calculator:
    def add(self, a: float, b: float) -> float:
        return a + b

    def subtract(self, a: float, b: float) -> float:
        return a - b

    def multiply(self, a: float, b: float) -> float:
        return a * b

    def divide(self, a: float, b: float) -> float:
        if b == 0:
            raise ZeroDivisionError("Cannot divide by zero")
        return a / b

    def is_prime_number(self, n: int) -> bool:
        if n < 2:
```

```
        return False
    if n == 2:
        return True
    if n % 2 == 0:
        return False

    i = 3
    while i * i <= n:
        if n % i == 0:
            return False
        i += 2
    return True
```

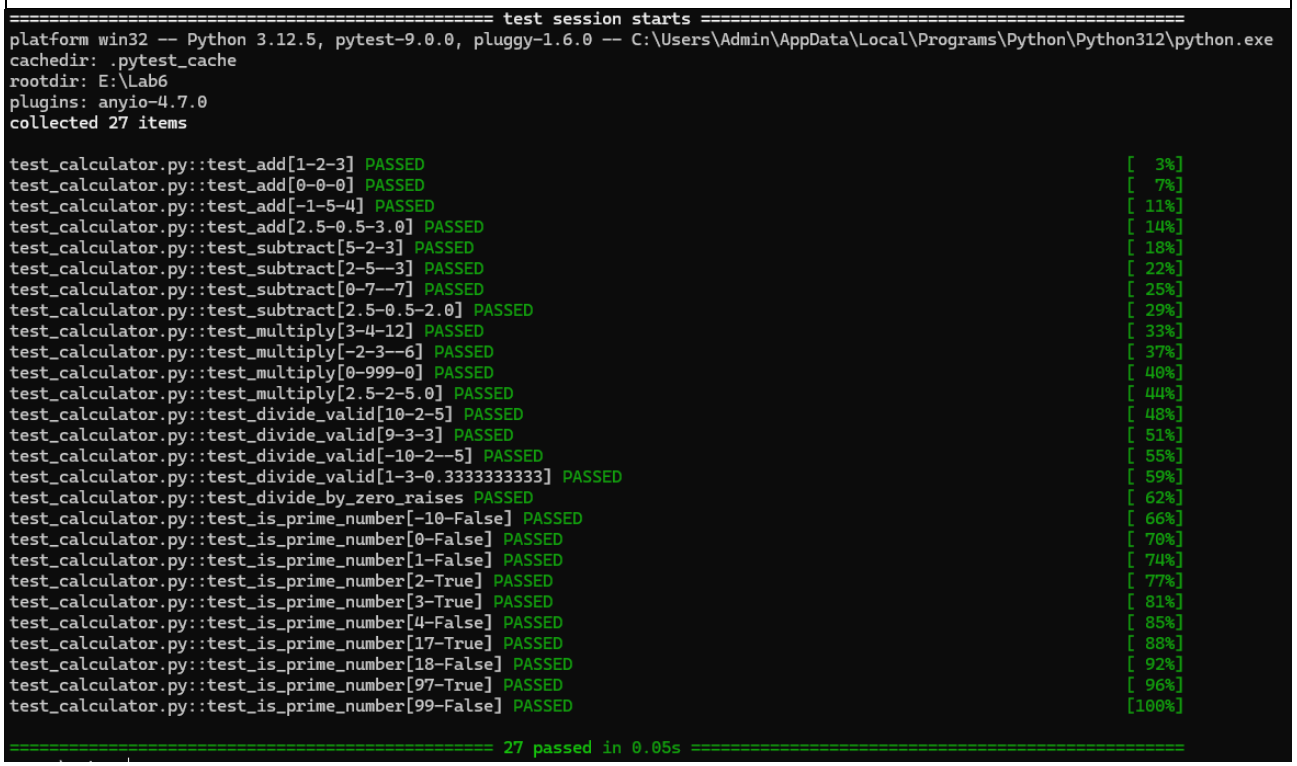
test_calculator.py

```
import pytest
from calculator import Calculator

@pytest.fixture
def calc():
    return Calculator()

@pytest.mark.parametrize(
    "a,b,expected",
    [
        (1, 2, 3),
        (0, 0, 0),
        (-1, 5, 4),
        (2.5, 0.5, 3.0),
    ],
```

```
)
def test_add(calc, a, b, expected):
    assert calc.add(a, b) == expected



@pytest.mark.parametrize(
    "a,b,expected",
    [
        (5, 2, 3),
        (2, 5, -3),
        (0, 7, -7),
        (2.5, 0.5, 2.0),
    ],
)
def test_subtract(calc, a, b, expected):
    assert calc.subtract(a, b) == expected

@pytest.mark.parametrize(
    "a,b,expected",
    [
        (3, 4, 12),
        (-2, 3, -6),
        (0, 999, 0),
        (2.5, 2, 5.0),
    ],
)
def test_multiply(calc, a, b, expected):
    assert calc.multiply(a, b) == expected

@pytest.mark.parametrize(
```

```
    "a,b,expected",
    [
        (10, 2, 5),
        (9, 3, 3),
        (-10, 2, -5),
        (1, 3, 0.3333333333),
    ],
)
def test_divide_valid(calc, a, b, expected):
    assert calc.divide(a, b) == pytest.approx(expected)


def test_divide_by_zero_raises(calc):
    with pytest.raises(ZeroDivisionError):
        calc.divide(10, 0)


@pytest.mark.parametrize(
    "n,expected",
    [
        (-10, False),
        (0, False),
        (1, False),
        (2, True),
        (3, True),
        (4, False),
        (17, True),
        (18, False),
        (97, True),
        (99, False),
    ],
)
```

```
def test_is_prime_number(calc, n, expected):
    assert calc.is_prime_number(n) is expected
```



```
============================================= test session starts =============================================
platform win32 -- Python 3.12.5, pytest-9.0.0, pluggy-1.6.0 -- C:\Users\Admin\AppData\Local\Programs\Python\Python312\python.exe
cachedir: .pytest_cache
rootdir: E:\Lab6
plugins: anyio-4.7.0
collected 27 items

test_calculator.py::test_add[1-2-3] PASSED                                                          [  3%]
test_calculator.py::test_add[0-0-0] PASSED                                                          [  7%]
test_calculator.py::test_add[-1-5-4] PASSED                                                         [ 11%]
test_calculator.py::test_add[2.5-0.5-3.0] PASSED                                                    [ 14%]
test_calculator.py::test_subtract[5-2-3] PASSED                                                     [ 18%]
test_calculator.py::test_subtract[2-5--3] PASSED                                                    [ 22%]
test_calculator.py::test_subtract[0-7--7] PASSED                                                    [ 25%]
test_calculator.py::test_subtract[2.5-0.5-2.0] PASSED                                               [ 29%]
test_calculator.py::test_multiply[3-4-12] PASSED                                                    [ 33%]
test_calculator.py::test_multiply[-2-3--6] PASSED                                                   [ 37%]
test_calculator.py::test_multiply[0-999-0] PASSED                                                   [ 40%]
test_calculator.py::test_multiply[2.5-2-5.0] PASSED                                                 [ 44%]
test_calculator.py::test_divide_valid[10-2-5] PASSED                                                [ 48%]
test_calculator.py::test_divide_valid[9-3-3] PASSED                                                 [ 51%]
test_calculator.py::test_divide_valid[-10-2--5] PASSED                                              [ 55%]
test_calculator.py::test_divide_valid[1-3-0.3333333333] PASSED                                      [ 59%]
test_calculator.py::test_divide_by_zero_raises PASSED                                               [ 62%]
test_calculator.py::test_is_prime_number[-10-False] PASSED                                          [ 66%]
test_calculator.py::test_is_prime_number[0-False] PASSED                                            [ 70%]
test_calculator.py::test_is_prime_number[1-False] PASSED                                            [ 74%]
test_calculator.py::test_is_prime_number[2-True] PASSED                                             [ 77%]
test_calculator.py::test_is_prime_number[3-True] PASSED                                             [ 81%]
test_calculator.py::test_is_prime_number[4-False] PASSED                                            [ 85%]
test_calculator.py::test_is_prime_number[17-True] PASSED                                            [ 88%]
test_calculator.py::test_is_prime_number[18-False] PASSED                                           [ 92%]
test_calculator.py::test_is_prime_number[97-True] PASSED                                            [ 96%]
test_calculator.py::test_is_prime_number[99-False] PASSED                                           [100%]

============================================== 27 passed in 0.05s ==============================================
```