**Министерство науки и высшего образования Российской Федерации** **Федеральное государственное бюджетное образовательное учреждение высшего образования** **«Московский государственный технический университет** **имени Н.Э. Баумана**

**(национальный исследовательский университет)»**

**(МГТУ им. Н.Э. Баумана)**

**Факультет «Информатика и системы управления»**

**Кафедра ИУ5 «Системы обработки информации и управления»**

Лабораторная работа №5

по дисциплине «Базовые компоненты интернет-технологий»

Выполнил:

студент группы ИУ5-33Б

Рассказов Н.Д.

Проверил:

Гапанюк Ю. Е.

2022 г.

1. **Описание задания.**
2. Выберите любой фрагмент кода из лабораторных работ 1 или 2 или 3-4.
3. Модифицируйте код таким образом, чтобы он был пригоден для модульного тестирования.
4. Разработайте модульные тесты. В модульных тестах необходимо применить следующие технологии:
   * TDD - фреймворк (не менее 3 тестов).
   * BDD - фреймворк (не менее 3 тестов).
   * Создание Mock-объектов (необязательное дополнительное задание).
5. Текст программы.

test1.feature

Feature: My first feature file using radish

In order to test my awesome software

I need an awesome BDD tool like radish

to test my software.

Scenario: Test get\_roots function

Given I have biquadrate equation with coefficients 1 and 0 and 10

When I find the roots of it

Then I expect the result to be ""

test2.feature

Feature: My first feature file using radish

In order to test my awesome software

I need an awesome BDD tool like radish

to test my software.

Scenario: Test get\_roots function

Given I have biquadrate equation with coefficients 1 and 0 and -16

When I find the roots of it

Then I expect the result to be "-2, 2"

test3.feature

Feature: My first feature file using radish

In order to test my awesome software

I need an awesome BDD tool like radish

to test my software.

Scenario: Test get\_roots function

Given I have biquadrate equation with coefficients -4 and 16 and 0

When I find the roots of it

Then I expect the result to be "-2, 0, 2"

main.py

*import* math

*import* pytest

def get\_roots(*a*, *b*, *c*):

*'''*

*Вычисление корней квадратного уравнения*

*Args:*

*a (float): коэффициент А*

*b (float): коэффициент B*

*c (float): коэффициент C*

*Returns:*

*list[float]: Список корней*

*'''*

    result = []

    D = *b*\**b* - 4\**a*\**c*

*if* D == 0.0:

        root = -*b* / (2.0\**a*)

*if* root > 0:

            root1 = math.sqrt(root)

            root2 = -math.sqrt(root)

            result.append(root1)

            result.append(root2)

*elif* root == 0:

            result.append(root)

*elif* D > 0.0:

        sqD = math.sqrt(D)

        first\_root = (-*b* + sqD) / (2.0\**a*)

        second\_root = (-*b* - sqD) / (2.0\**a*)

*if* first\_root > 0:

            root1 = math.sqrt(first\_root)

            root2 = -math.sqrt(first\_root)

            result.append(root1)

            result.append(root2)

*elif* first\_root == 0:

            result.append(first\_root)

*if* second\_root > 0:

            root3 = math.sqrt(second\_root)

            root4 = -math.sqrt(second\_root)

            result.append(root3)

            result.append(root4)

*elif* second\_root == 0:

            result.append(second\_root)

*return* result

class TestRoots:

    def test\_0roots(*self*):

*assert* len(get\_roots(1, 0, 10)) == 0

    def test\_2roots(*self*):

        a = get\_roots(1, 0, -16)

*assert* (len(a) == 2) and (2 in a) and (-2 in a)

    def test\_3roots(*self*):

        a = get\_roots(-4, 16, 0)

*assert* (len(a) == 3) and (0 in a) and (2 in a) and (-2 in a)

    def test\_4roots(*self*):

        a = get\_roots(1, -5, 6)

*assert* (len(a) == 4) and (math.sqrt(3) in a) and (math.sqrt(3) in a)\

            and (math.sqrt(2) in a) and (-math.sqrt(2) in a)

radish/step.py

*import* math

*from* radish *import* given, when, then

def get\_roots(*a*, *b*, *c*):

*'''*

*Вычисление корней квадратного уравнения*

*Args:*

*a (float): коэффициент А*

*b (float): коэффициент B*

*c (float): коэффициент C*

*Returns:*

*list[float]: Список корней*

*'''*

    result = []

    D = *b*\**b* - 4\**a*\**c*

*if* D == 0.0:

        root = -*b* / (2.0\**a*)

*if* root > 0:

            root1 = math.sqrt(root)

            root2 = -math.sqrt(root)

            result.append(root1)

            result.append(root2)

*elif* root == 0:

            result.append(root)

*elif* D > 0.0:

        sqD = math.sqrt(D)

        first\_root = (-*b* + sqD) / (2.0\**a*)

        second\_root = (-*b* - sqD) / (2.0\**a*)

*if* first\_root > 0:

            root1 = math.sqrt(first\_root)

            root2 = -math.sqrt(first\_root)

            result.append(root1)

            result.append(root2)

*elif* first\_root == 0:

            result.append(first\_root)

*if* second\_root > 0:

            root3 = math.sqrt(second\_root)

            root4 = -math.sqrt(second\_root)

            result.append(root3)

            result.append(root4)

*elif* second\_root == 0:

            result.append(second\_root)

*return* result

@given("I have biquadrate equation with coefficients {number1:g} and {number2:g} and {number3:g}")

def have\_numbers(*step*, *number1*, *number2*, *number3*):

*step*.context.number1 = *number1*

*step*.context.number2 = *number2*

*step*.context.number3 = *number3*

@when("I find the roots of it")

def sum\_numbers(*step*):

*step*.context.result = sorted(get\_roots(

*step*.context.number1, *step*.context.number2, *step*.context.number3))

@then("I expect the result to be {result:QuotedString}")

def expect\_result(*step*, *result*):

    lst = *result*.split(", ")

*if* lst[0] == "" and len(lst) == 1:

        lst = []

*assert* len(*step*.context.result) == len(lst)

*if* len(lst) == 0:

*return*

    lst = list(map(float, lst))

*assert* *step*.context.result == lst

1. Экранные формы с примерами выполнения программы.



