Защищено:
Гапанюк Ю.Е.

"__"_____**2023** г.

Демонстрация:
Фень Н.Т.

"__"_____**2023** г.

Отчет по лабораторной работе № 3 по курсу Парадигмы и конструкции языков программирования

Тема работы: " Модульное тестирование в Python. "

5
(количество листов)
Вариант № **1**

ИСПОЛНИТЕЛЬ:

студент группы ИУ5-52Б _____
(подпись)

Фень Н.Т.

"__"_____**2023** г.

Москва, МГТУ - **2023**

СОДЕРЖАНИЕ

1. Описание задания

Разработайте бота для Telegram. Бот должен реализовывать конечный автомат из трех состояний.

Разработайте простого бота для Telegram. Бот должен использовать функциональность создания кнопок.

2. Текст программы

lab1module.py:

```
import sys
import cmath
#Модифицированный код ЛР1

# Функция для ввода коэфицентов и проверки на ошибки
def get_coefficient(prompt, coef_name):
    while True:
        try:
            coef = float(input(prompt))
            return coef
        except ValueError:
            print(f"Некорректное значение для коэффициента {coef_name}. Повторите
ввод.")


# Функция для вычисления корней уравнения
def solve_biquadratic(a, b, c):
    discriminant = b ** 2 - 4 * a * c

    if discriminant > 0:
        root1 = cmath.sqrt((-b + cmath.sqrt(discriminant)) / (2 * a))
        root2 = -root1
        root3 = cmath.sqrt((-b - cmath.sqrt(discriminant)) / (2 * a))
        root4 = -root3
        return root1, root2, root3, root4
    elif discriminant == 0:
        root = cmath.sqrt(-b / (2 * a))
        return root, -root
    else:
        return ()


# Функция поиска действительных корней
def findTrueRoots(roots):
    trueRoots = []
    for root in roots:
        # Проверка на действительность и на дубликатность ( 0 и -0)
        if (root.imag == 0) and (str(root.real) != "-0.0"):
            trueRoots.append(root.real)
    return trueRoots


def printTrueRoots(trueRoots):
    if len(trueRoots) == 4:
        print(
            f"Уравнение имеет четыре действительных корня: {trueRoots[0]},
{trueRoots[1]}, {trueRoots[2]}, {trueRoots[3]}")
    elif len(trueRoots) == 3:
        print(f"Уравнение имеет три действительных корня: {trueRoots[0]},
{trueRoots[1]}, {trueRoots[2]}")
    elif len(trueRoots) == 2:
        print(f"Уравнение имеет два действительных корня: {trueRoots[0]},
{trueRoots[1]}")
    elif len(trueRoots) == 1:
```

```
        print(f"Уравнение имеет один действительный корень: {trueRoots[0]}")
    else:
        print("Уравнение не имеет действительных корней.")


def main():
    # Проверка ввода из консоли и получение аргументов
    if len(sys.argv) == 4:
        try:
            a = float(sys.argv[1])
            b = float(sys.argv[2])
            c = float(sys.argv[3])
        except ValueError:
            print("Некорректные коэффициенты в командной строке. Пожалуйста, введите
их с клавиатуры.")
            a = get_coefficient("Введите коэффициент A: ", "A")
            b = get_coefficient("Введите коэффициент B: ", "B")
            c = get_coefficient("Введите коэффициент C: ", "C")
    else:
        a = get_coefficient("Введите коэффициент A: ", "A")
        b = get_coefficient("Введите коэффициент B: ", "B")
        c = get_coefficient("Введите коэффициент C: ", "C")
    # Получение корней
    roots = solve_biquadratic(a, b, c)
    # Вывод ответа
    trueRoots = findTrueRoots(roots)
    printTrueRoots(trueRoots)


if __name__ == "__main__":
    main()
```

lab3.py:

```
# Тесты TDD

from lab1Module import findTrueRoots, solve_biquadratic
import math

def test_solve_biquadratic_real_roots():
    # Test case with real roots
    roots = solve_biquadratic(1, -3, 2)

    expected_roots = [math.sqrt(2), -math.sqrt(2), 1.0, -1.0]  # roots are (1, 2)
and their negatives
    assert findTrueRoots(roots) == expected_roots

def test_solve_biquadratic_complex_roots():
    # Test case with complex roots
    roots = solve_biquadratic(1, 1, 1)
    expected_roots = ()  # no real roots
    assert roots == expected_roots

def test_find_true_roots():
    # Test case for find_true_roots function
    roots = [1 + 0j, 2 + 0j, 0 + 1j, -1 - 1j]  # mix of real and complex roots
    true_roots = findTrueRoots(roots)
    expected_true_roots = [1, 2]
    assert true_roots == expected_true_roots
```

BDD.txt:

#Тесты BDD

Функция: Решение биквадратных уравнений

Сценарий: допустимый ввод коэффициента.
Учитывая, что у меня запущена программа
Когда я ввожу действительный коэффициент A
Тогда программа должна принять ввод

Сценарий: решить уравнение с положительным дискриминантом.
Учитывая, что у меня есть коэффициенты A, B и C
Когда я ввожу эти коэффициенты
Тогда программа должна вывести правильные корни