



МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
“Национальный исследовательский университет ИТМО”

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ и КОМПЬЮТЕРНОЙ ТЕХНИКИ

0000-0?

ଭାବୁ ଭାବୁ ଭାବୁ
ଭାବୁ ଭାବୁ ଭାବୁ
ଭାବୁ ଭାବୁ ଭାବୁ

ЛАБОРАТОРНАЯ РАБОТА №2

по дисциплине

“Тестирование программного обеспечения”

Вариант: 881.

ПИиКТ

Работу выполнил:

Студент группы Р3311

Лектор:

VI-ой семестр

Клименков Сергей Викторович

Практик:

геньевич

Птицын Максим Евгеньевич

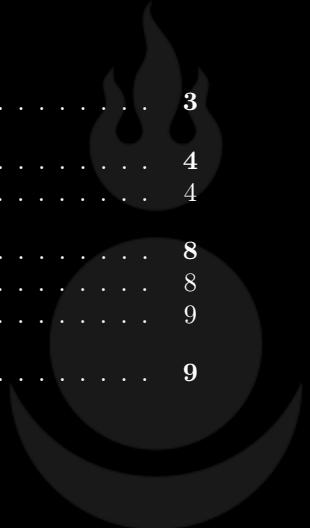
— 10 —

Птицын Максим Евгеньевич

г. Санкт-Петербург
2025 г.

Содержимое

1 Текст задания	3
2 Выполнение	4
2.1 Графики тригонометрических функций (с анализом эквивалентности)	4
3 Дополнительные задания	8
3.1 Реализация	8
3.2 Результат	9
4 Вывод	9



ТПО.

0000-0?

000 046 000
000 051 017

010 000 030

020 020 006

000 002 001

000 000 000

000 000 000

000 000 000

000 000 000

000 000 000

000 000 000

000 000 000

000 000 000

000 000 000

000 000 000

000 000 000

000 000 000

000 000 000

000 000 000

000 000 000

000 000 000

000 000 000

000 000 000

000 000 000

000 000 000

000 000 000

000 000 000

000 000 000

000 000 000

000 000 000

ULTRAVIOLENCE.

ПИиКТ

и-ой семестр

20/21

YK



00 000 000
00 000 000
00 000 000

00 000 000
00 000 000
00 000 000

00 000 000
00 000 000
00 000 000

00 000 000
00 000 000
00 000 000

00 000 000
00 000 000
00 000 000

00 000 000
00 000 000
00 000 000

00 000 000
00 000 000
00 000 000

00 000 000
00 000 000
00 000 000

00 000 000
00 000 000
00 000 000

1 Текст задания

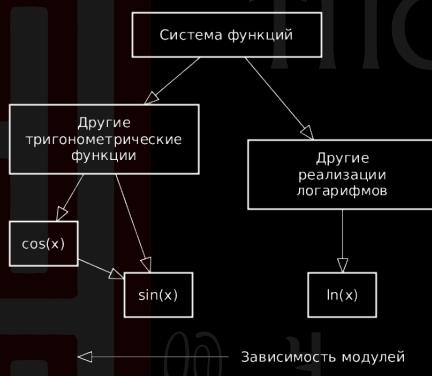
Провести интеграционное тестирование программы, осуществляющей вычисление системы функций (в соответствии с вариантом).

Функция:

$$f(x) = \begin{cases} (((((\tan(x) \cdot \cos(x)) + (\cot(x) + \sec(x))) + \csc(x)) \cdot \csc(x)) + \sec(x)) & \text{if } x \leq 0 \\ (((((\log_{10}(x) - \log_{10}(x)) \cdot (\log_2(x) - \log_3(x))) \cdot (\frac{\log_3(x) - \log_5(x)}{\log_{10}(x)})^3) + (\frac{\ln(x) \cdot \log_{10}(x)}{\log_2(x)})) & \text{if } x > 0 \end{cases} \quad (1)$$

Правила выполнения работы:

1. Все составляющие систему функции (как тригонометрические, так и логарифмические) должны быть выражены через базовые (тригонометрическая зависит от варианта; логарифмическая - натуральный логарифм).
2. Структура приложения, тестируемого в рамках лабораторной работы, должна выглядеть следующим образом (пример приведён для базовой тригонометрической функции $\sin(x)$):



3. Обе "базовые" функции (в примере выше - $\sin(x)$ и $\ln(x)$) должны быть реализованы при помощи разложения в ряд с задаваемой погрешностью. Использовать тригонометрические / логарифмические преобразования для упрощения функций ЗАПРЕЩЕНО.
4. Для КАЖДОГО модуля должны быть реализованы табличные заглушки. При этом, необходимо найти область допустимых значений функций, и, при необходимости, определить взаимозависимые точки в модулях.
5. Разработанное приложение должно позволять выводить значения, выдаваемое любым модулем системы, в csv файл вида «Х, Результаты модуля (Х)», позволяющее произвольно менять шаг наращивания Х. Разделитель в файле csv можно использовать произвольный.

Порядок выполнения работы:

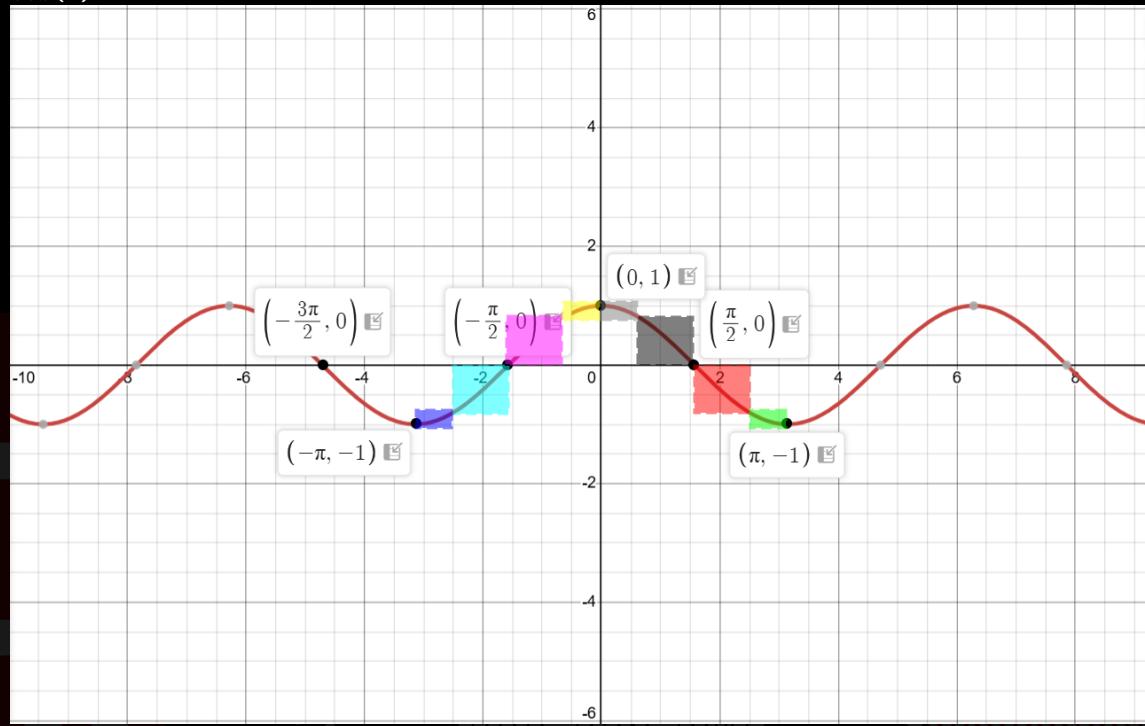
1. Разработать приложение, руководствуясь приведёнными выше правилами.
2. С помощью JUNIT4 разработать тестовое покрытие системы функций, проведя анализ эквивалентности и учитывая особенности системы функций. Для анализа особенностей системы функций и составляющих ее частей можно использовать сайт <https://www.wolframalpha.com/>.
3. Собрать приложение, состоящее из заглушек. Провести интеграцию приложения по 1 модулю, с обоснованием стратегии интеграции, проведением интеграционных тестов и контролем тестового покрытия системы функций.

2 Выполнение

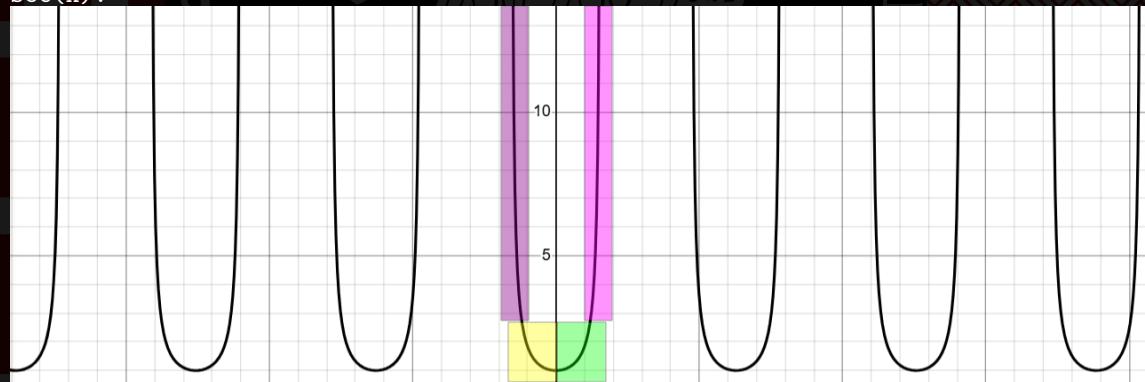
Ссылка на репозиторию: [GitHub](#)

2.1 Графики тригонометрических функций (с анализом эквивалентности)

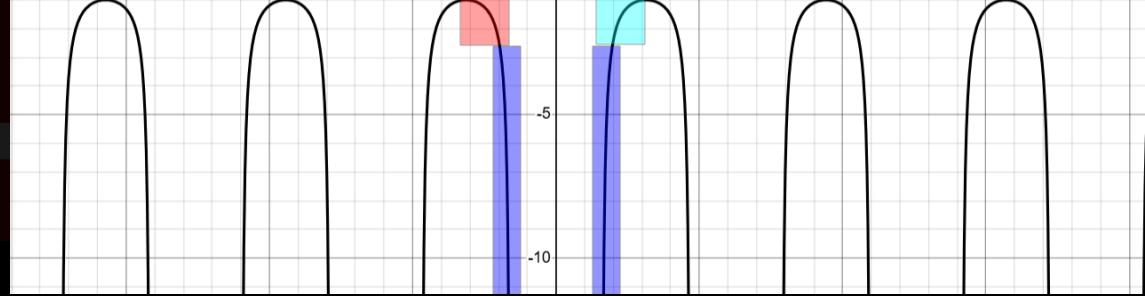
1. $\cos(x)$:

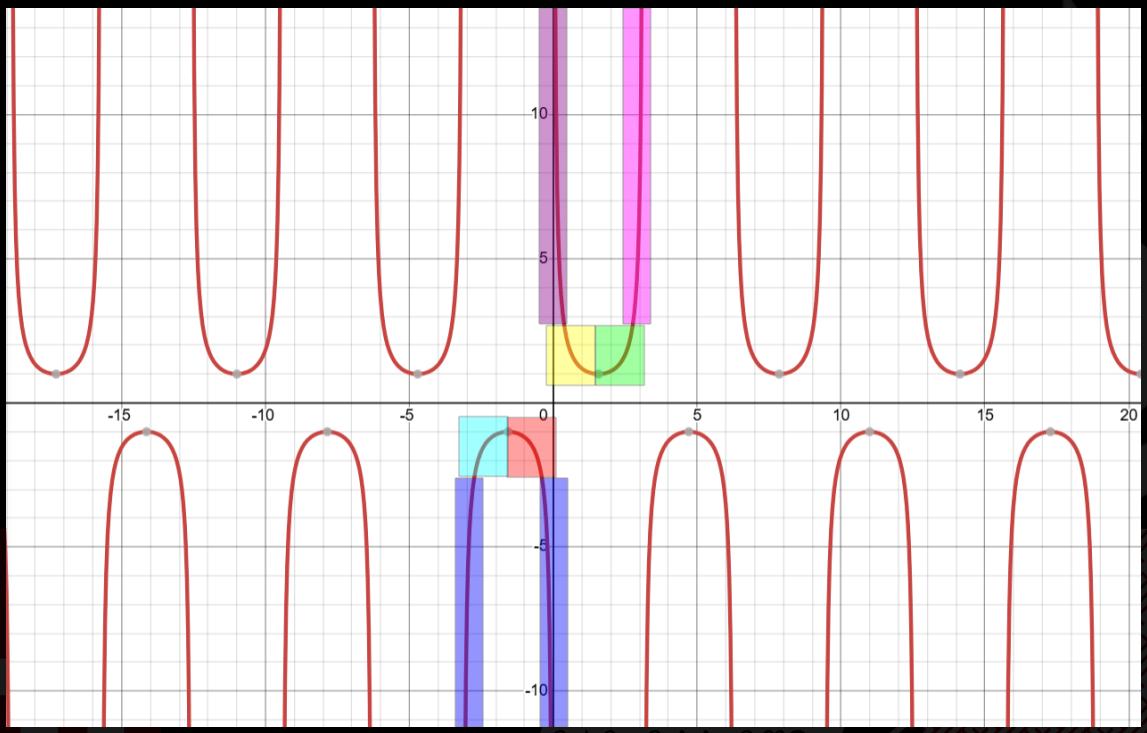


2. $\sec(x)$:

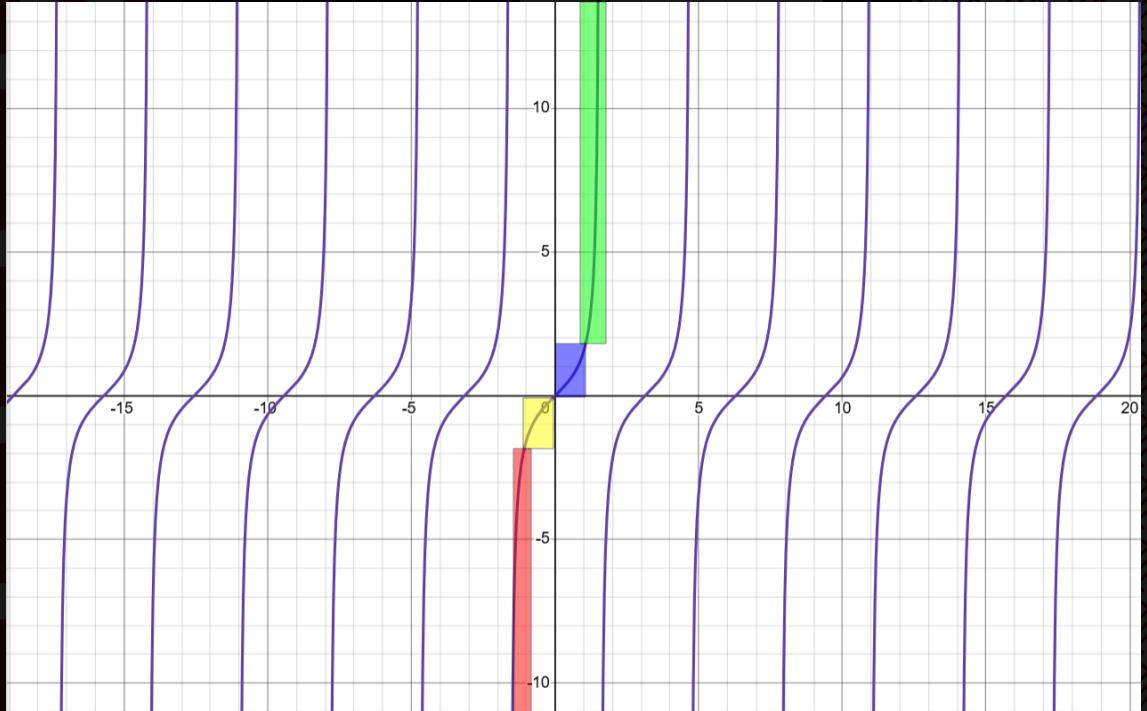


3. $\csc(x)$:

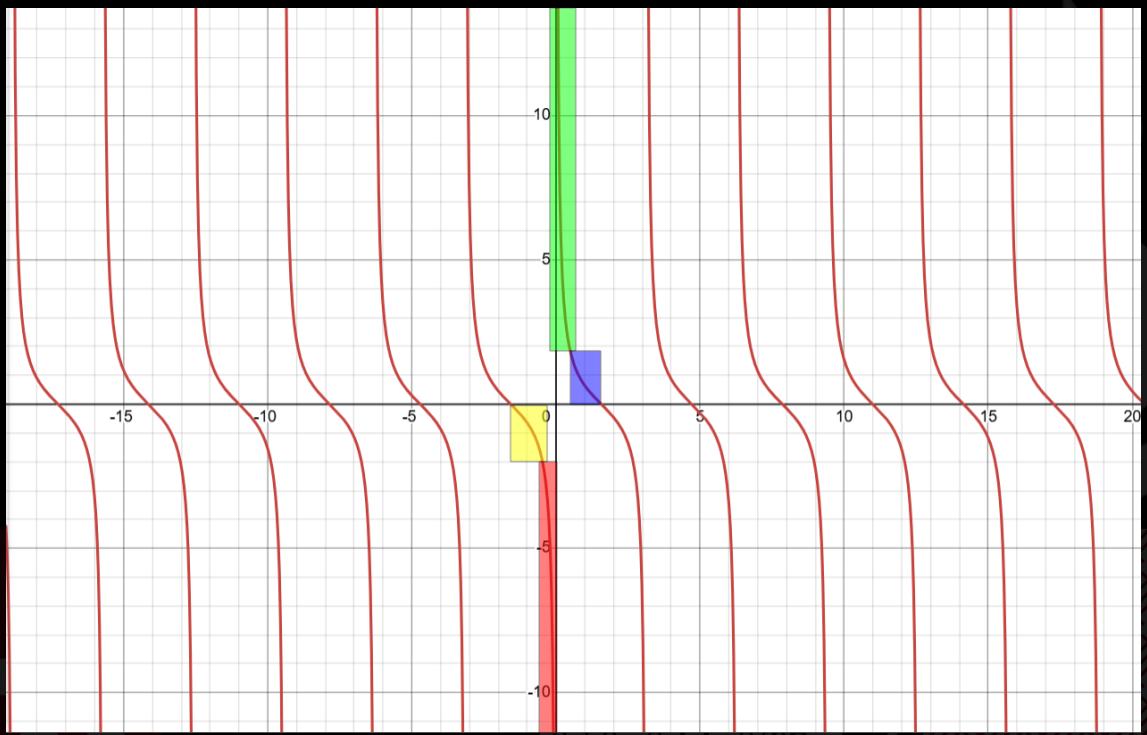




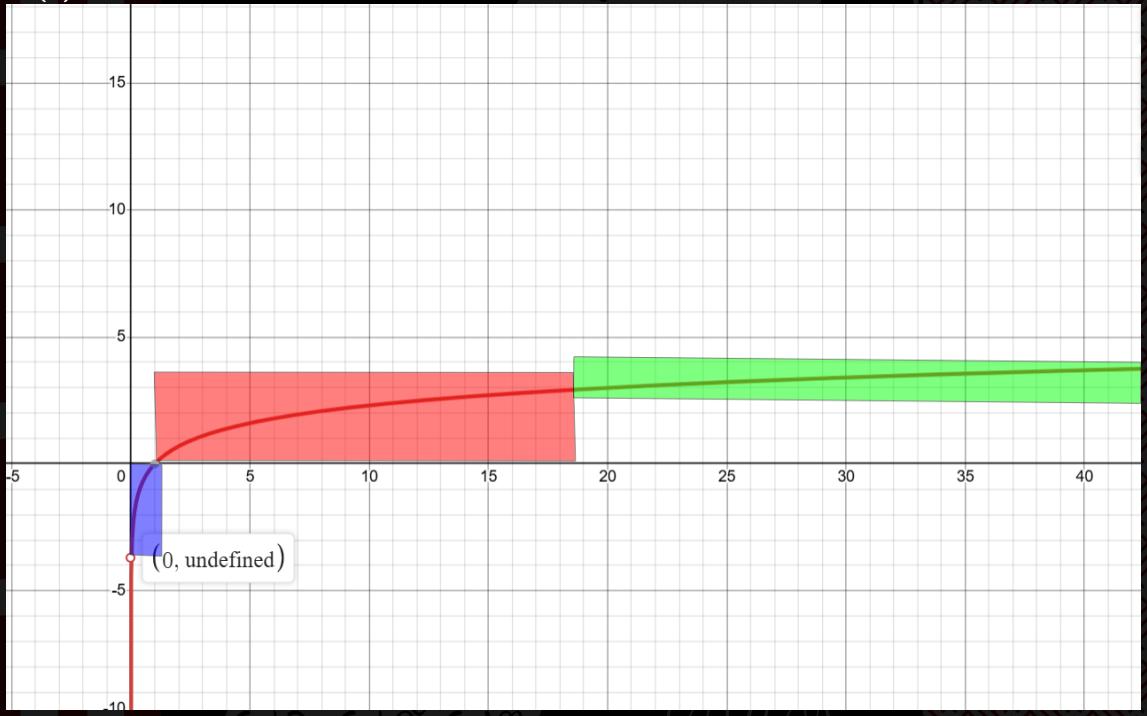
4. $\tan(x)$:



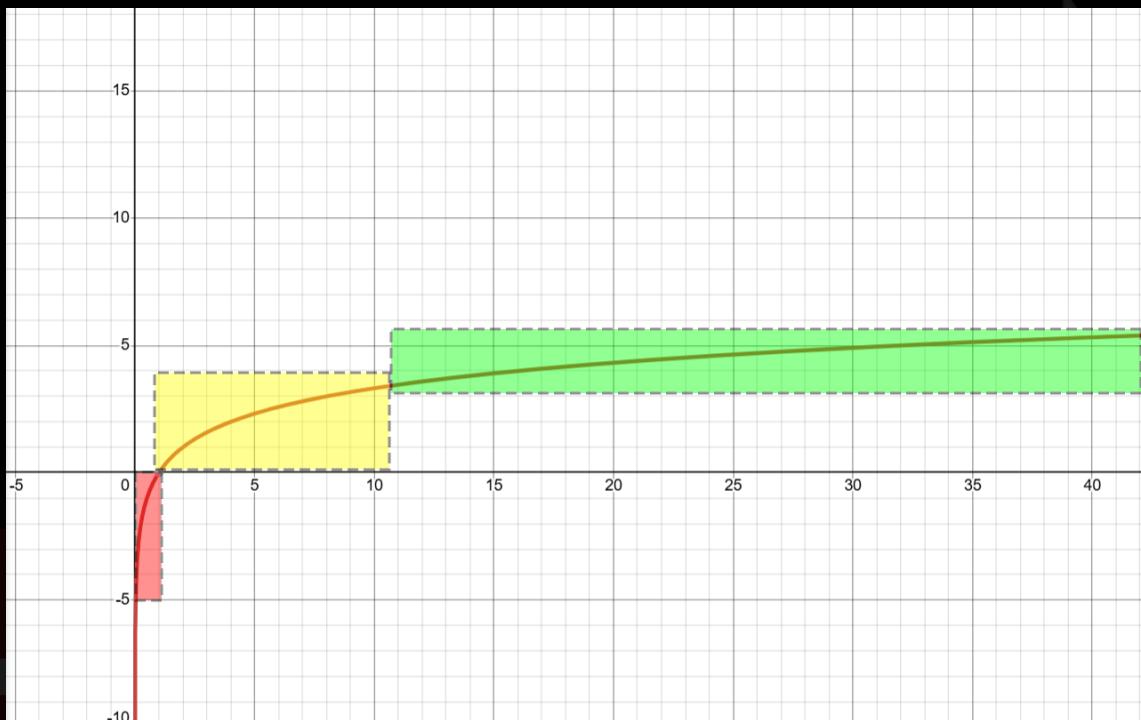
5. $\cot(x)$:



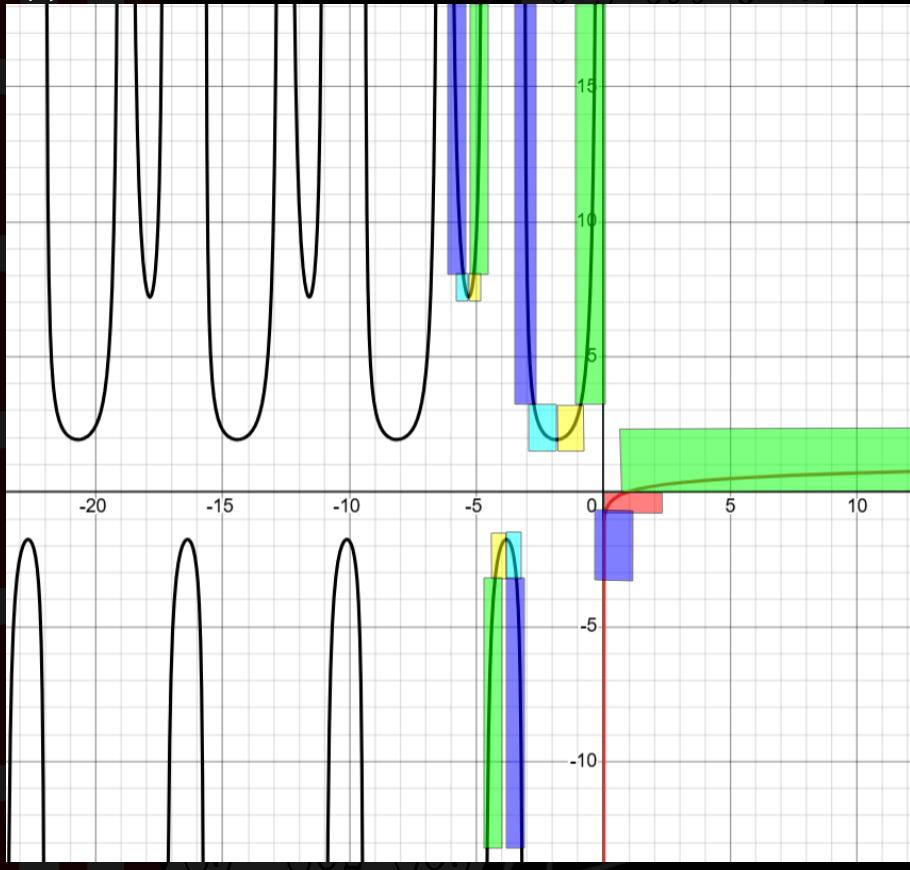
6. $\ln(x)$:



7. $\log(x)$:



8. $f(x)$:



Один из методов теста:

lab2/LogarithmIntegrationTest.java

```
@Test  
void shouldCallLn() {
```

```

        final BaseNLogarithm logarithm = new BaseNLogarithm(5, spyLn);
        logarithm.calculate(new BigDecimal(993), new BigDecimal("0.001"));
        verify(spyLn, atLeastOnce()).calculate(any(BigDecimal.class), any(BigDecimal.class));
    }

    @Test
    void shouldCalculateWithMockLn() {
        final BigDecimal arg = new BigDecimal(1488);
        when(mockLn.calculate(eq(new BigDecimal(1488)), any(BigDecimal.class))).thenReturn(new BigDecimal("4.5389687"));
        when(mockLn.calculate(eq(new BigDecimal(5)), any(BigDecimal.class))).thenReturn(new BigDecimal("1.648721270700127"));

        final BaseNLogarithm log5 = new BaseNLogarithm(5, mockLn);
        final BigDecimal expected = new BigDecimal("4.5389687");
        assertEquals(expected, log5.calculate(arg, PRECISION));
    }
}

```

3 Дополнительные задания.

1. Использовать шпионы для создания частичной заглушки.
2. Реализовать проверку тестового покрытия.

3.1 Реализация

LogarithmIntegrationTest.java

```

@Mock
private NaturalLogarithm mockLn;
@Spy
private NaturalLogarithm spyLn;

@Test
void shouldCallLn() {
    final BaseNLogarithm logarithm = new BaseNLogarithm(5, spyLn);
    logarithm.calculate(new BigDecimal(993), new BigDecimal("0.001"));
    verify(spyLn, atLeastOnce()).calculate(any(BigDecimal.class), any(BigDecimal.class));
}

```

Шпион зовет сам метод, в отличие от полной заглушки.

build.gradle.kts

```

jacoco {
    toolVersion = "0.8.12"
    reportsDirectory = layout.buildDirectory.dir("customJacocoReportDir")
}

tasks.test {
    extensions.configure(JacocoTaskExtension::class) {
        setDestinationFile(file(layout.buildDirectory.file("jacoco/jacoco.exec")))
    }
    finalizedBy(tasks.jacocoTestReport)
}

tasks.jacocoTestReport {
    dependsOn(tasks.test)
}

```

```
reports {
    xml.required = false
    csv.required = false
    html.required = true
    html.outputLocation = file(layout.buildDirectory.dir("reports/coverage"))
}
finalizedBy(tasks.jacocoTestCoverageVerification)
}

tasks.jacocoTestCoverageVerification {
    violationRules {
        rule {
            limit {
                minimum = "0.85".toBigDecimal()
            }
        }
    }
}
```

3.2 Результат

4 Вывод

В рамке этой лабораторной работы я написал модульные тесты по нескольким доменам. Также я упражнял написание .yml файлов для настройки GitHub CI.