**Тема:** *Исследование программного кода на предмет ошибок и отклонения от алгоритма*

**Формат:** Индивидуальный проект

**Цель задания:**

1. Научиться внимательно анализировать код, сверяя его с описанием алгоритма или требованиями.
2. Выявлять различные типы ошибок: синтаксические, логические, ошибки в условиях, граничные случаи.
3. Научиться проверять корректность кода с помощью тестовых случаев.
4. Развивать критическое мышление при оценке чужого (и своего) кода.

**Теоретическая часть (Краткие ответы, ~1 стр.)**

1. **Типы ошибок в коде:**
   * Дайте краткое определение и приведите **по одному примеру** для следующих типов ошибок:
     + Синтаксическая ошибка
     + Логическая ошибка
     + Ошибка в условии (неправильное условие)
     + Ошибка обработки граничных случаев
     + Отклонение от алгоритма (код делает не то, что описано)
2. **Методы поиска ошибок:**
   * Перечислите 3 основных метода, которые вы будете использовать в этом задании (напр., *Визуальный анализ кода*, *Сравнение с описанием алгоритма*, *Ручное тестирование на конкретных примерах*).

**Практическая часть (Анализ кода)**

**Задача:**  
Написать функцию **is\_perfect\_square(n)**, которая возвращает **True**, если целое число **n** является точным квадратом (т.е. **n = k \* k** для некоторого целого **k**), и **False** в противном случае.

* **Алгоритм (Спецификация):**
  1. Если **n < 0**, вернуть **False** (отрицательные числа не могут быть квадратами).
  2. Если **n == 0** или **n == 1**, вернуть **True (0\*0=0, 1\*1=1).**
  3. Для **n > 1**:
     + Использовать **бинарный поиск** в диапазоне от **2**до **n // 2** (т.к. корень из **n**не может быть больше **n/2** при**n>1**).
     + Найти целое **mid** такое, что **mid \* mid == n.** Если найдено, вернуть **True**.
     + Если бинарный поиск завершился неудачей, вернуть **False**.

**Дан код студента: файл test5.py**

**Задания:**

1. **Визуальный анализ и сравнение с алгоритмом:**
   * Прочитайте код **строку за строкой**. Сверьте каждую часть кода с описанием алгоритма выше.
   * Выпишите **номера строк**, где код, по вашему мнению, содержит **ошибку или отклонение от алгоритма**. Для каждой найденной проблемы:
     + Кратко опишите **суть ошибки/отклонения**.
     + Объясните, **почему это ошибка/отклонение** (ссылаясь на спецификацию).
     + *Пример ответа:* "Строка X: Ошибка типа [логическая/условия/граничная/отклонение]. Описание: ... Почему: ..."
2. **Поиск мнимых "ошибок":**
   * Есть ли в коде места, которые *выглядят* подозрительно на первый взгляд, но на самом деле **являются корректными**? Если да, укажите номера строк и объясните, почему код здесь корректен.
3. **Ручное тестирование:**
   * Составьте **таблицу тестовых случаев**. Включите:
     + Обязательные случаи: отрицательные числа, 0, 1, обычные квадраты (4, 9, 16), обычные НЕ квадраты (2, 3, 5, 7, 15).
     + *Критические случаи:* Большой квадрат (напр., 1000000), число, близкое к квадрату (напр., 15, 17 для квадрата 16).
   * Для каждого теста предскажите **Ожидаемый результат** (согласно спецификации) и **Фактический результат** (как отработает *данный* код).
   * Выделите те тесты, где **Фактический результат не совпадет с Ожидаемым** (они и выявят ошибки).
4. **Исправление кода:**
   * Напишите **исправленную версию функции** **is\_perfect\_square**, устранив все обнаруженные ошибки.
   * Кратко (1-2 предложения) поясните **ключевые изменения**.
5. **Анализ и выводы:**
   * Какая **наиболее серьезная ошибка** была в коде? Почему?
   * Какой **тип ошибки** (из Теоретической части) встречался чаще всего в этом примере?
   * Какой **тестовый случай** оказался самым эффективным для поиска ошибки?

**Форма отчета(файл .doc и .py):**

1. **Теоретическая часть:** Краткие ответы на вопросы.
2. **Практическая часть:**
   * Список найденных ошибок/отклонений с описанием (Пункт 1).
   * Список корректных, но подозрительных мест (Пункт 2).
   * Таблица тестов (Пункт 3).
   * Исправленный код с пояснениями (Пункт 4).
   * Ответы на вопросы анализа (Пункт 5).