1.

```
fun getTwoDigitNumber(): Int {
    while (true) {
        print("Введите двузначное число: ")
        val input = readLine() ?: ""
        try {
            val number = input.toInt()
            require(number in 10..99) { "Число должно быть двузначным" }
            return number
        } catch (e: NumberFormatException) {
            println("Неверный формат ввода. Попробуйте ещё раз.")
        }
    }
}

fun analyzeNumber(number: Int): Quadruple<Int, Int, Int, Int> {
    val tens = number / 10
    val units = number % 10
    val sum = tens + units
    val product = tens * units
    return Quadruple(tens, units, sum, product)
}

fun main() {
    val number = getTwoDigitNumber()
    val (tens, units, sum, product) = analyzeNumber(number)
    println("Число десятков: $tens")
    println("Число единиц: $units")
    println("Сумма цифр: $sum")
    println("Произведение цифр: $product")
}

data class Quadruple<out A, out B, out C, out D>(val first: A, val second: B,
val third: C, val fourth: D)
```

2.

```
fun main() {
    // Ввод трехзначного числа
    println("Введите трехзначное число:")
    val input = readLine()

    // Проверка, что введенное значение является трехзначным числом
    if (input != null && input.length == 3 && input.all { it.isDigit() }) {
        val number = input.toInt()

        // Извлечение цифр
        val hundreds = number / 100
        val tens = (number / 10) % 10
        val units = number % 10

        // Вычисление суммы и произведения цифр
        val sum = hundreds + tens + units
        val product = hundreds * tens * units

        // Вывод результатов
        println("Число десятков: $tens")
        println("Число единиц: $units")
        println("Сумма цифр: $sum")
        println("Произведение цифр: $product")
    } else {
        println("Ошибка: введите корректное трехзначное число.")
    }
}
```

```
3. fun main() {
    // Ввод числа и степени
    println("Введите число:")
    val numberInput = readLine()

    println("Введите степень:")
    val exponentInput = readLine()

    // Проверка на корректность ввода
    if (numberInput != null && exponentInput != null) {
        try {
            val number = numberInput.toDouble() // Преобразуем ввод в число
            val exponent = exponentInput.toInt() // Преобразуем ввод в
степень

            // Вычисление результата
            val result = Math.pow(number, exponent.toDouble())

            // Вывод результата
            println("$number в степени $exponent равно $result")
        } catch (e: NumberFormatException) {
            println("Ошибка: введите корректные числа.")
        }
    } else {
        println("Ошибка: ввод не может быть пустым.")
    }
}

4. import kotlin.math.sqrt

fun main() {
    println("Введите число для вычисления его квадратного корня:")

    // Чтение ввода от пользователя
    val input = readLine()

    // Проверка на null и преобразование строки в число
    if (input != null) {
        try {
            val number = input.toDouble()

            // Проверка на неотрицательное число
            if (number < 0) {
                println("Ошибка: Квадратный корень из отрицательного числа не
существует.")
            } else {
                // Вычисление квадратного корня
                val squareRoot = sqrt(number)
                println("Квадратный корень из $number равен $squareRoot")
            }
        } catch (e: NumberFormatException) {
            println("Ошибка: Введите корректное число.")
        }
    } else {
        println("Ошибка: Ввод не может быть пустым.")
    }
}
```