1.

```
import kotlin.random.Random

fun main() {
    val randomNumbers = List(10) { Random.nextInt(1, 101) }
    println(randomNumbers)
}
```

2.

```
fun analyzeString(text: String) {
    val vowels = "aeiouAEIOU"
    var vowelCount = 0
    var consonantCount = 0

    for (char in text) {
        if (char.isLetter()) { // Проверяем, является ли символ буквой
            if (char in vowels) {
                vowelCount++
            } else {
                consonantCount++
            }
        }
    }

    println("Гласных: $vowelCount")
    println("Согласных: $consonantCount")
}

fun main() {
    val text = "Привет, мир!"
    analyzeString(text)

    val text2 = "Hello, World!"
    analyzeString(text2)


    val text3 = "12345" // Test with no letters
    analyzeString(text3)

}
```

3.

```
import java.util.Scanner

fun main() {
    val scanner = Scanner(System.`in`)
    val exchangeRates = mapOf(
        "USD" to 1.0, // Доллар США (базовая валюта)
        "EUR" to 0.85, // Евро
        "GBP" to 0.73, // Британский фунт
        "JPY" to 145.0 // Японская иена
        // Добавьте другие валюты и курсы по мере необходимости
    )

    println("Доступные валюты: ${exchangeRates.keys}")

    print("Введите сумму: ")
    val amount = scanner.nextDouble()

    print("Введите исходную валюту (например, USD): ")
    val fromCurrency = scanner.next().uppercase()

    print("Введите валюту, в которую нужно конвертировать (например, EUR): ")
```

```
    val toCurrency = scanner.next().uppercase()


    if (!exchangeRates.containsKey(fromCurrency) ||
!exchangeRates.containsKey(toCurrency)) {
        println("Ошибка: одна или обе валюты не поддерживаются.")
        return
    }


    val fromRate = exchangeRates[fromCurrency]!!
    val toRate = exchangeRates[toCurrency]!!

    val convertedAmount = amount * (toRate / fromRate)

    println("$amount $fromCurrency = $convertedAmount $toCurrency")

}
```

4.

```
fun isAnagram(str1: String, str2: String): Boolean {
    // Приводим строки к нижнему регистру и удаляем не буквенные символы
    val normalizedStr1 = str1.lowercase().filter { it.isLetter() }
    val normalizedStr2 = str2.lowercase().filter { it.isLetter() }

    // Проверяем длину строк. Если длины не равны, то строки не могут быть
анаграммами
    if (normalizedStr1.length != normalizedStr2.length) {
        return false
    }


    return
normalizedStr1.toCharArray().sortedArray().contentEquals(normalizedStr2.toCha
rArray().sortedArray())

}


// Более эффективное решение с использованием HashMap
fun isAnagramHashMap(str1: String, str2: String): Boolean {
    val normalizedStr1 = str1.lowercase().filter { it.isLetter() }
    val normalizedStr2 = str2.lowercase().filter { it.isLetter() }

    if (normalizedStr1.length != normalizedStr2.length) {
        return false
    }

    val charCounts = mutableMapOf<Char, Int>()

    for (char in normalizedStr1) {
        charCounts[char] = charCounts.getOrDefault(char, 0) + 1
    }

    for (char in normalizedStr2) {
        charCounts[char] = charCounts.getOrDefault(char, 0) - 1
        if (charCounts[char]!! < 0) {
            return false // Если какой-то символ встречается чаще во второй
строке
        }
    }

    return charCounts.all { it.value == 0 }
```

```
}




fun main() {
    println(isAnagram("listen", "silent"))     // Output: true
    println(isAnagram("hello", "world"))      // Output: false
    println(isAnagram("Elbow", "Below"))       // Output: true (ignores case)
    println(isAnagram("Dormitory", "Dirty room")) // Output: true (ignores
spaces)


    println(isAnagramHashMap("listen", "silent"))     // Output: true
    println(isAnagramHashMap("hello", "world"))      // Output: false

}
```

5.

```
fun isPrime(n: Int): Boolean {
    if (n <= 1) return false
    if (n <= 3) return true
    if (n % 2 == 0 || n % 3 == 0) return false

    var i = 5
    while (i * i <= n) {
        if (n % i == 0 || n % (i + 2) == 0) return false
        i += 6
    }
    return true
}



fun printPrimesUpToN(n: Int) {
    for (i in 2..n) {
        if (isPrime(i)) {
            print("$i ")
        }
    }
    println()
}


fun main() {
    val n = 50
    println("Простые числа до $n:")
    printPrimesUpToN(n)
}
```

6.

```
fun sortStrings(strings: Array<String>): Array<String> {
    return strings.sortedArray() // Используем встроенную функцию
sortedArray()
}


// Альтернативный вариант с использованием sorted() и toTypedArray()
fun sortStrings2(strings: Array<String>): Array<String> {
    return strings.sorted().toTypedArray()
```

```
}



fun main() {
    val strings = arrayOf("banana", "apple", "orange", "grape")
    val sortedStrings = sortStrings(strings)

    println("Отсортированный массив:")
    println(sortedStrings.contentToString()) // Выводим отсортированный
массив

    // Демонстрация второго варианта:
    val strings2 = arrayOf("zebra", "yak", "xray", "wombat")
    val sortedStrings2 = sortStrings2(strings2)
    println(sortedStrings2.contentToString())

}
```

7.

```
fun changeCase(text: String): String {
    val result = StringBuilder() // Используем StringBuilder для эффективного
построения строки
    for (char in text) {
        if (char.isLetter()) {
            result.append(
                if (char.isUpperCase()) char.lowercaseChar() else
char.uppercaseChar()
            )
        } else {
            result.append(char) // Небуквенные символы остаются без изменений
        }
    }
    return result.toString()
}

fun main() {
    val text = "Привет, Мир!"
    val changedText = changeCase(text)
    println(changedText) // Вывод: пРИВЕТ, мИР!

    val text2 = "Hello, World!"
    val changedText2 = changeCase(text2)
    println(changedText2) // Вывод: hELLO, wORLD!

    val text3 = "123abcXYZ"
    val changedText3 = changeCase(text3)
    println(changedText3) // Вывод: 123ABCxyz


}
```

8.

```
import kotlin.random.Random

fun main() {
    val randomNumber = Random.nextInt(1, 101) // Генерируем случайное число
от 1 до 100
    var guess: Int? = null
    var attempts = 0


    println("Добро пожаловать в игру 'Угадай число'!")
    println("Я загадал число от 1 до 100. Попробуйте угадать.")
```

```
    while (guess != randomNumber) {
        print("Введите ваше предположение: ")
        try {
            guess = readLine()?.toIntOrNull()

            if (guess == null) {
                println("Некорректный ввод. Пожалуйста, введите число.")
                continue
            }



            attempts++


            if (guess < randomNumber) {
                println("Загаданное число больше.")
            } else if (guess > randomNumber) {
                println("Загаданное число меньше.")
            }


        } catch (e: NumberFormatException) {
            println("Некорректный ввод. Пожалуйста, введите число.")
        }

    }

    println("Поздравляю! Вы угадали число $randomNumber за $attempts
попыток.")

}
```

9.

```
import kotlin.random.Random

fun generatePassword(length: Int, useDigits: Boolean = true, useLetters:
Boolean = true, useSpecialChars: Boolean = true): String {
    if (length <= 0) return ""

    val charPool = buildList {
        if (useDigits) addAll('0'..'9')
        if (useLetters) {
            addAll('a'..'z')
            addAll('A'..'Z')
        }
        if (useSpecialChars) addAll("!@#$%^&*()_+-
=[]{}|;':\",./<>?".toCharArray().toList())
    }


    if (charPool.isEmpty()) return ""


    return (1..length)
        .map { charPool.random(Random) }
        .joinToString("")
}


fun main() {
    val password = generatePassword(12)
```

```
    println(password)

    val password2 = generatePassword(8, useSpecialChars = false)
    println(password2)

    val password3 = generatePassword(6, useDigits = false, useLetters =
false) // Only special characters
    println(password3)


}
```

10.

```
fun findLongestWord(text: String): String {
    require(text.isNotBlank()) { "Входная строка не может быть пустой или
состоять только из пробелов." }

    val words = text.split(Regex("\s+|[.,!?;:\"'()]")) // Разделяем строку на
слова, игнорируя знаки препинания и несколько пробелов
        .filter { it.isNotBlank() } // Удаляем пустые строки, которые могут
появиться из-за двойных пробелов или знаков препинания


    var longestWord = ""
    for (word in words) {
        if (word.length > longestWord.length) {
            longestWord = word
        }
    }


    return longestWord
}


fun main() {
    val text1 = "Это предложение с длинными и короткими словами."
    val longestWord1 = findLongestWord(text1)
    println("Самое длинное слово: $longestWord1") // Вывод: предложение

    val text2 = "Привет, мир! Как дела?"
    val longestWord2 = findLongestWord(text2)
    println("Самое длинное слово: $longestWord2") // Вывод: Привет

    val text3 = "Два длинных слова: автомобиль, велосипед"
    val longestWord3 = findLongestWord(text3)
    println("Самое длинное слово: $longestWord3") // Вывод: автомобиль


    // Проверка require:
    //val text4 = ""
    //val longestWord4 = findLongestWord(text4) // Выбросит
IllegalArgumentException
}
```