Лабораторная работа №4

**1.**

```
fun main() {
    fun getDayOfWeek(day: Int): String {
        return when (day) {
            1 -> "Понедельник"
            2 -> "Вторник"
            3 -> "Среда"
            4 -> "Четверг"
            5 -> "Пятница"
            6 -> "Суббота"
            7 -> "Воскресенье"
            else -> "Некорректный номер дня"
        }
    }
    val dayNumber = 3
    val dayOfWeek = getDayOfWeek(dayNumber)
    println("День недели: $dayOfWeek")
}
```

**2.**

```
fun main() {
    fun determineTriangleType(a: Double, b: Double, c: Double): String {
        return when {
            a <= 0 || b <= 0 || c <= 0 -> "Некорректные длины сторон"
            a + b <= c || a + c <= b || b + c <= a -> "Не треугольник"
            a == b && b == c -> "Равносторонний треугольник"
            a == b || b == c || a == c -> "Равнобедренный треугольник"
            else -> "Разносторонний треугольник"
        }
    }
    val sideA = 3.0
    val sideB = 4.0
    val sideC = 5.0
```

```
    val triangleType = determineTriangleType(sideA, sideB, sideC)
    println("Тип треугольника: $triangleType")
}
```

**3.**

```
fun main() {
    fun getGrade(score: Int): String {
        return when (score) {
            in 90..100 -> "Оценка: A"
            in 80..89 -> "Оценка: B"
            in 70..79 -> "Оценка: C"
            in 60..69 -> "Оценка: D"
            in 0..59 -> "Оценка: F"
            else -> "Некорректное значение"
        }
    }
    val score = 85
    val grade = getGrade(score)
    println(grade)
}
```

**4**.

```
fun main() {
    fun getTimeOfDay(hour: Int): String {
        return when (hour) {
            in 0..5 -> "Ночь"
            in 6..11 -> "Утро"
            in 12..17 -> "День"
            in 18..23 -> "Вечер"
            else -> "Некорректное время"
        }
    }
    val hour = 14
    val timeOfDay = getTimeOfDay(hour)
```

```
    println(timeOfDay)
}
```

**5.**

```
fun main() {
    fun determineSign(number: Int): String {
        return when {
            number > 0 -> "Положительное"
            number < 0 -> "Отрицательное"
            else -> "Ноль"
        }
    }
    val number = -7
    val sign = determineSign(number)
    println(sign)
}
```

**6.**

```
fun main() {
    val randomNumber = Random.nextInt(1, 101) // Число от 1 до 100
    var guess: Int
    var attempts = 0
    println("Угадайте число от 1 до 100!")
    while (true) {
        print("Введите ваше предположение: ")
        guess = readLine()?.toIntOrNull() ?: continue
        attempts++
        when {
            guess < randomNumber -> println("Слишком маленькое число! Попробуйте снова."
  guess > randomNumber -> println("Слишком большое число! Попробуйте снова.")
            else -> {
                println("Поздравляем! Вы угадали число $randomNumber за $attempts попыток!")
                exitProcess(0)
            }
```

```
        }
    }
}
```

**7.**

```
fun main() {
    print("Введите строку: ")
    val inputString = readLine() ?: return
    val length = inputString.length
    println("Длина строки: $length")
}
```

**8.**

```
fun main() {
    println("Введите тип пищи (например, 'блюдо', 'десерт', 'напиток'):")
    val foodType = readLine()?.lowercase()
    val cookingTime = when (foodType) {
        "блюдо" -> "обычно 30-60 минут"
        "десерт" -> "обычно 15-30 минут"
        "напиток" -> "обычно 5-15 минут"
        else -> "Неизвестный тип пищи"
    }
    println("Время приготовления: $cookingTime")
}
```

**9**.

```
fun main() {
  println("Введите строку:")
    val inputString = readLine() ?: ""
    val length = inputString.length
    println("Длина введенной строки: $length")
}
```

**10.**

```
fun main() {
    println("Выберите способ оплаты (наличные, кредитная карта, PayPal):")
    val paymentMethod = readLine()
    when (paymentMethod) {
        "наличные" -> println("Вы выбрали оплату наличными.")
        "кредитная карта" -> println("Вы выбрали оплату кредитной картой.")
        "PayPal" -> println("Вы выбрали оплату через PayPal.")
        else -> println("Неизвестный способ оплаты. Пожалуйста, попробуйте снова.")
    }
}
```

**11.**

```
fun determineCompatibleBloodTypes(bloodType: String): String {
    when (bloodType.uppercase()) {
        "A" -> return "A, O"
        "B" -> return "B, O"
        "AB" -> return "A, B, AB, O"
        "O" -> return "O"
        else -> return "Неизвестная группа крови"
    }
}
fun main() {
    println("Введите группу крови (A, B, AB, O):")
    val inputBloodType = readLine() ?: ""
 val compatibleBloodTypes = determineCompatibleBloodTypes(inputBloodType)
    println("Совместимые группы крови для переливания: $compatibleBloodTypes")
}
```

**12.**

```
fun getNationality(country: String): String {
    val nationalities = mapOf(
        "США" to "Американец/американка",
        "Россия" to "Россиянин/россиянка",
        "Япония" to "Японец/японка",
```

```
        "Германия" to "Немец/немка",
        "Франция" to "Француз/француженка",
        "Китай" to "Китаец/китаянка",
        "Великобритания" to "Британец/британка",
        "Италия" to "Итальянец/итальянка",
        "Испания" to "Испанец/испанка",
        "Канада" to "Канадец/канадка",
        "Австралия" to "Австралиец/австралийка",
        "Бразилия" to "Бразилец/бразильянка",
        "Индия" to "Индиец/индианка", // Обратите внимание на различие
        "Мексика" to "Мексиканец/мексиканка"
        // Добавьте другие страны и национальности по мере необходимости
    )
    return nationalities[country] ?: "Национальность не найдена для этой страны"
}
fun main() {
    println("Введите название страны:")
    val countryName = readLine() ?: ""
    val nationality = getNationality(countryName)
  println("Национальность: $nationality")
}
```

**13**.

```
fun getErrorMessage(errorCode: Int): String {
    return when (errorCode) {
        100 -> "Ошибка авторизации"
        200 -> "Ошибка сети"
        300 -> "Ошибка сервера"
        400 -> "Неверный запрос"
        401 -> "Доступ запрещен"
        404 -> "Не найдено"
        500 -> "Внутренняя ошибка сервера"
        else -> "Неизвестная ошибка ($errorCode)" // Обработка неизвестных кодов
    }
```

```
}
fun main() {
    print("Введите код ошибки: ")
    val inputCode = readLine()?.toIntOrNull() // Безопасное чтение целого числа
    if (inputCode != null) {
        val errorMessage = getErrorMessage(inputCode)
        println(errorMessage)
    } else {
        println("Некорректный ввод. Введите число.")
    }
}
```