

1.

```
fun calculator() {  
    while (true) {  
        println("Простой калькулятор")  
        println("1. Сложение")  
        println("2. Вычитание")  
        println("3. Умножение")  
        println("4. Деление")  
        println("5. Выход")  
  
        print("Выберите операцию (1-5): ")  
        val choice = readLine()  
  
        if (choice == "5") {  
            println("До свидания!")  
            break  
        }  
  
        if (choice !in listOf("1", "2", "3", "4")) {  
            println("Неверный выбор")  
            continue  
        }  
  
        print("Введите первое число: ")  
        val num1 = readLine()?.toDoubleOrNull() ?: continue  
  
        print("Введите второе число: ")  
        val num2 = readLine()?.toDoubleOrNull() ?: continue  
  
        val result = when (choice) {  
            "1" -> num1 + num2
```

```

        "2" -> num1 - num2

        "3" -> num1 * num2

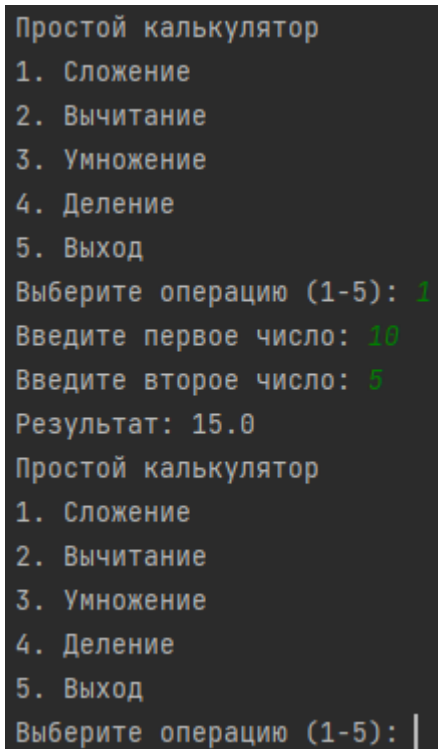
        "4" -> if (num2 != 0.0) num1 / num2 else {
            println("Ошибка: деление на ноль!")
            continue
        }

        else -> continue
    }

    println("Результат: $result")
}

fun main() {
    calculator()
}

```



```

Простой калькулятор
1. Сложение
2. Вычитание
3. Умножение
4. Деление
5. Выход
Выберите операцию (1-5): 1
Введите первое число: 10
Введите второе число: 5
Результат: 15.0
Простой калькулятор
1. Сложение
2. Вычитание
3. Умножение
4. Деление
5. Выход
Выберите операцию (1-5): |

```

2.

```

fun isPalindrome(word: String): Boolean {
    val cleanWord = word.lowercase()

```

```

        return cleanWord == cleanWord.reversed()
    }

    fun main() {
        print("Введите слово: ")
        val word = readLine() ?: return
        if (isPalindrome(word)) {
            println("'${word}' является палиндромом")
        } else {
            println("'${word}' не является палиндромом")
        }
    }
}

```

```

Введите слово: казак
'казак' является палиндромом

Process finished with exit code 0

```

3.1

```

fun calculatePoints(wins: Int, draws: Int, losses: Int): Int {
    return (wins * 3) + (draws * 1) + (losses * 0)
}

fun main() {
    val wins = 5
    val draws = 3
    val losses = 2
    val points = calculatePoints(wins, draws, losses)
    println("Количество очков команды: $points")
}

```

```

Количество очков команды: 18

Process finished with exit code 0

```

3.2

```

fun findMinNumber(numbers: List<Int>): Int? {
    return numbers.minOrNull()
}

```

```

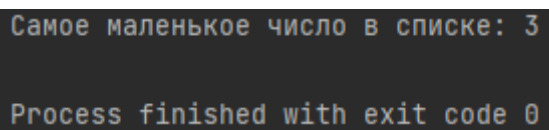
fun main() {
    // Пример списка чисел
}

```

```
val numbers = listOf(10, 5, 8, 3, 12)
```

```
val minNumber = findMinNumber(numbers)
```

```
if (minNumber != null) {  
    println("Самое маленькое число в списке: $minNumber")  
} else {  
    println("Список пуст.")  
}  
}
```



```
Самое маленькое число в списке: 3  
Process finished with exit code 0
```

3.3

```
fun areNumbersEqual(num1: Int, num2: Int): Boolean {  
    return num1 == num2  
}
```

```
fun main() {  
    val reader = java.util.Scanner(System.`in`)  
  
    println("Введите первое число:")  
    val number1 = reader.nextInt()  
  
    println("Введите второе число:")  
    val number2 = reader.nextInt()  
  
    val result = areNumbersEqual(number1, number2)  
    println("Числа равны: $result")  
}
```

```
Введите первое число:  
5  
Введите второе число:  
5  
Числа равны: true  
  
Process finished with exit code 0
```

4.

```
import kotlin.random.Random
```

```
fun main() {  
    val deck = createDeck()  
    val playerHand = mutableListOf<Int>()  
    val dealerHand = mutableListOf<Int>()  
  
    // Начальная раздача карт  
    playerHand.add(drawCard(deck))  
    playerHand.add(drawCard(deck))  
    dealerHand.add(drawCard(deck))  
    dealerHand.add(drawCard(deck))  
  
    // Игровой цикл  
    var playerTurn = true  
    while (true) {  
        println("Ваши карты: $playerHand, сумма: ${calculateHandValue(playerHand)}")  
        println("Карты дилера: [{dealerHand[0]}, ?]")  
  
        if (playerTurn) {  
            println("Хотите взять еще карту? (да/нет)")  
            val response = readLine()  
            if (response.equals("да", ignoreCase = true)) {  
                playerHand.add(drawCard(deck))  
                if (calculateHandValue(playerHand) > 21) {
```

```

        println("Ваши карты: $playerHand, сумма: ${calculateHandValue(playerHand)}")
        println("Вы превысили 21! Вы проиграли.")
        return
    }
} else {
    playerTurn = false
}
} else {
    while (calculateHandValue(dealerHand) < 17) {
        dealerHand.add(drawCard(deck))
    }
    break
}
}

```

// Подведение итогов

```

println("Ваши карты: $playerHand, сумма: ${calculateHandValue(playerHand)}")
println("Карты дилера: $dealerHand, сумма: ${calculateHandValue(dealerHand)}")

```

```

val playerScore = calculateHandValue(playerHand)

```

```

val dealerScore = calculateHandValue(dealerHand)

```

```

when {
    dealerScore > 21 -> println("Дилер превысил 21! Вы выиграли!")
    playerScore > dealerScore -> println("Вы выиграли!")
    playerScore < dealerScore -> println("Вы проиграли!")
    else -> println("Ничья!")
}
}

```

```

fun createDeck(): MutableList<Int> {
    return (1..10).flatMap { card -> List(4) { card } }.toMutableList()
}

```

```
}
```

```
fun drawCard(deck: MutableList<Int>): Int {  
    val randomIndex = Random.nextInt(deck.size)  
    return deck.removeAt(randomIndex)  
}
```

```
fun calculateHandValue(hand: List<Int>): Int {  
    return hand.sum()  
}
```

```
Ваши карты: [9, 9], сумма: 18  
Карты дилера: [7, ?]  
Хотите взять еще карту? (да/нет)  
нет  
Ваши карты: [9, 9], сумма: 18  
Карты дилера: [7, ?]  
Ваши карты: [9, 9], сумма: 18  
Карты дилера: [7, 5, 2, 10], сумма: 24  
Дилер превысил 21! Вы выиграли!  
  
Process finished with exit code 0
```