```
//Francis Miguel Kuhl Schweitzer Júnior
//Lucas Ferreira Neto
//Ana Luisa Ribeiro
//Aline Duarte Sutil
import kotlin.random.Random

fun main() {

    while (true) {
        var portAv = 10 //mínimo 10% de l*c && máximo 30% l*c
        var cruzad = 1 //mínimo 1% de l*c && máximo 5% l*c
        var reboc = 2  //mínimo 2% de l*c && máximo  10% de l*c
        var l = 10
        var c = 10
        var numTiros = 15

        println("Batalha Naval")

        // Tabuleiro
        while (true) {
            println("Deseja alterar o tamanho do tabuleiro? ")
            var lerTabu = readln().trim()

            if (lerTabu.equals("sim", ignoreCase = true)) {
                while (true) {
                    println("Qual tamanho você vai querer para as linhas? ")
                    l = readln().toInt()
                    println("Qual tamanho você vai querer para as colunas? ")
                    c = readln().toInt()

                    if ((c >= 5 && l >= 5) && (c <= 50 && l <= 50)) {
                        println("Tamanho alterado")
                        break
                    } else {
                        println("Tamanho inválido, o tamanho permitido é de 5x5 até 50x50")
                    }
                }
                break
            } else if (lerTabu.equals("não", ignoreCase = true) || lerTabu.equals("nao",
ignoreCase = true)) {
                println("Mantendo o tabuleiro padrão")
```

```
            break
        } else {
            println("Valor inválido. Digite 'sim' ou 'não'.")
        }
    }

    val agua = "\uD83D\uDCA7"
    val arrayTabuleiro = Array(l) { Array(c) { "$agua" } }
    val tabJogador = Array(l) { Array(c) { "$agua" } }
    println("Tabuleiro Padrão: Linha: $l  Coluna: $c")

    val totalnavios = portAv + reboc + cruzad
    val casasTabu = l * c

    //Barcos

    while (true) {
        println("\nDeseja alterar a quantidade de navios? Selecione uma opção:")
        println("1 - Porta Aviões (Padrão: $portAv) [Min: 10% | Max: 30% do tabuleiro]")
        println("2 - Cruzadores   (Padrão: $cruzad) [Min: 1%  | Max: 5%  do tabuleiro]")
        println("3 - Rebocadores  (Padrão: $reboc) [Min: 2%  | Max: 45% do tabuleiro]")
        println("4 - Não alterar e continuar")
        val opcao = readln().toIntOrNull()


        when (opcao) {
            1 -> {
                val minPortAv = (0.10 * casasTabu).toInt()
                val maxPortAv = (0.30 * casasTabu).toInt()
                println("Digite a quantidade de Porta Aviões (Min: $minPortAv | Max:
$maxPortAv):")
                val valor = readln().toIntOrNull()
                if (valor != null && valor >= minPortAv && valor <= maxPortAv) {
                    portAv = valor
                    println("Porta Aviões definido para $portAv")
                } else {
                    println("Valor inválido! Deve estar entre $minPortAv e $maxPortAv")
                }
            }

            2 -> {
```

```
                val minCruzad = (0.01 * casasTabu).toInt()
                val maxCruzad = (0.05 * casasTabu).toInt()
                println("Digite a quantidade de Cruzadores (Min: $minCruzad | Max:
$maxCruzad):")
                val valor = readln().toIntOrNull()
                if (valor != null && valor >= minCruzad && valor <= maxCruzad) {
                    cruzad = valor
                    println("Cruzadores definido para $cruzad")
                } else {
                    println("Valor inválido! Deve estar entre $minCruzad e $maxCruzad")
                }
            }

            3 -> {
                val minReboc = (0.02 * casasTabu).toInt()
                val maxReboc = (0.10 * casasTabu).toInt()
                println("Digite a quantidade de Rebocadores (Min: $minReboc | Max:
$maxReboc):")
                val valor = readln().toIntOrNull()
                if (valor != null && valor >= minReboc && valor <= maxReboc) {
                    reboc = valor
                    println("Rebocadores definido para $reboc")
                } else {
                    println("Valor inválido! Deve estar entre $minReboc e $maxReboc")
                }
            }

            4 -> {
                println("Mantendo os valores atuais de navios.")
                break
            }

            else -> {
                println("Opção inválida! Escolha entre 1, 2, 3 ou 4.")
            }
        }
    }
    // Quantidade de tiros
    while (true) {
        println("\nO número atual de tiros é $numTiros")
        println("Deseja alterar o número de tiros?")
```

```
    println("1 - Sim")
    println("2 - Não")
    val opcaoTiros = readln().toIntOrNull()

    when (opcaoTiros) {
        1 -> {
            val minTiros = totalnavios
            val maxTiros = casasTabu
            println("Digite o novo número de tiros (Min: $minTiros | Max: $maxTiros):")
            val novoTiros = readln().toIntOrNull()
            if (novoTiros != null && novoTiros >= minTiros && novoTiros <= maxTiros) {
                numTiros = novoTiros
                println("Número de tiros definido para $numTiros")
                break
            } else {
                println("Valor inválido! Deve estar entre $minTiros e $maxTiros")
            }
        }

        2 -> {
            println("Mantendo o número de tiros padrão: $numTiros")
            break
        }

        else -> {
            println("Opção inválida! Escolha 1 para Sim ou 2 para Não.")
        }
    }
}

// Posicionar navios
fun posicionarNavios(tab: Array<Array<String>>, qtd: Int, simbolo: String) {
    var colocados = 0
    while (colocados < qtd) {
        val linha = Random.nextInt(tab.size)
        val coluna = Random.nextInt(tab[0].size)

        if (tab[linha][coluna] == "$agua") {
            tab[linha][coluna] = simbolo
            colocados++
        }
```

```
    }
}

posicionarNavios(arrayTabuleiro, portAv, " ✈️ ")
posicionarNavios(arrayTabuleiro, cruzad, " 🚢 ")
posicionarNavios(arrayTabuleiro, reboc, " 🚤 ")

// Imprimir tabuleiro
fun imprimirTabuleiro(tab: Array<Array<String>>) {
    for (linha in tab) {
        println(linha.joinToString(" "))
    }
}

// Verificar oque o tiro acertou
fun verificarTiro(linha: Int, coluna: Int, tab: Array<Array<String>>): String {
    return when (tab[linha][coluna]) {
        " ✈️ " -> "Porta Aviões"
        " 🚢 " -> "Cruzador"
        " 🚤 " -> "Rebocador"
        else -> "Água"
    }
}

// Dar dica
fun darDica(linha: Int, coluna: Int, tab: Array<Array<String>>) {
    var dica = "M. Errou por muito, distância maior que 3."
    for (i in 1..3) {
        if (linha - i >= 0 && tab[linha - i][coluna] != "\uD83D\uDCA7") {
            dica = "$i casa(s) de distância acima"
            break
        }
        if (linha + i < tab.size && tab[linha + i][coluna] != "\uD83D\uDCA7") {
            dica = "$i casa(s) de distância abaixo"
            break
        }
        if (coluna - i >= 0 && tab[linha][coluna - i] != "\uD83D\uDCA7") {
            dica = "$i casa(s) de distância à esquerda"
            break
        }
```

```
        if (coluna + i < tab[0].size && tab[linha][coluna + i] != "\uD83D\uDCA7") {
            dica = "$i casa(s) de distância à direita"
            break
        }
    }
    println(dica)
}

// Processar o tiro
fun processarTiro(
    linhaTiro: Int,
    colunaTiro: Int,
    tab: Array<Array<String>>,
    tabJogador: Array<Array<String>>,
    pontos: Int
): Int {
    val resultado = verificarTiro(linhaTiro, colunaTiro, tab)

    when (resultado) {
        "Porta Aviões" -> {
            println("Alvo atingido! Porta aviões afundou!")
            println("Você ganhou +5 pontos")
            tab[linhaTiro][colunaTiro] = " 🔴 "
            tabJogador[linhaTiro][colunaTiro] = " 🔴 "
            return pontos + 5
        }

        "Cruzador" -> {
            println("Alvo atingido! Cruzador afundou!")
            println("Você ganhou +15 pontos")
            tab[linhaTiro][colunaTiro] = " 🔴 "
            tabJogador[linhaTiro][colunaTiro] = " 🔴 "
            return pontos + 15

        }

        "Rebocador" -> {
            println("Alvo atingido! Rebocador afundou!")
            println("Você ganhou +10 pontos")
            tab[linhaTiro][colunaTiro] = " 🔴 "
```

```
            tabJogador[linhaTiro][colunaTiro] = " 🔴 "
            return pontos + 10
        }

        "Água" -> {
            println("Água! Tentando novamente...")
            println("Seus pontos ainda são: $pontos")
            darDica(linhaTiro, colunaTiro, tab)
            tabJogador[linhaTiro][colunaTiro] = " 🟩 "
            return pontos
        }
    }
    return pontos
}

// Verificar se todos os navios foram afundados
fun todosNaviosAfundados(tab: Array<Array<String>>): Boolean {
    for (linha in tab) {
        for (celula in linha) {
            if (celula == " ✈️ " || celula == " 🚢 " || celula == " 🚤 ") {
                return false
            }
        }
    }
    return true
}

// Função do jogo
fun jogarTurno() {
    var pontos = 0
    var tirosRestantes = numTiros

    // Enquanto ainda houver tiros restantes
    while (tirosRestantes > 0) {
        println("\nTiros restantes: $tirosRestantes")
        imprimirTabuleiro(tabJogador)

        println("\nDigite as coordenadas para o seu tiro (linha e coluna):")
        val linha = readln().toIntOrNull()
        val coluna = readln().toIntOrNull()
```

```
        // Verifica se as coordenadas são válidas
        if (linha != null && coluna != null && linha in 0 until l && coluna in 0 until c) {

            // Verifica se o local já foi atingido
            if (tabJogador[linha][coluna] != "$agua") {
                println("Você já atirou aqui! Tente outro lugar.")
                continue
            }

            // Processa o tiro | atualiza os pontos
            pontos = processarTiro(linha, coluna, arrayTabuleiro, tabJogador, pontos)
            tirosRestantes--

            // Verifica se o jogador venceu
            if (todosNaviosAfundados(arrayTabuleiro)) {
                println("Parabéns! Você afundou todos os navios com $pontos pontos!")
                tirosRestantes = 1
                break
            }
        } else {
            println("Coordenadas inválidas! Tente novamente.")
        }
    }
    if (tirosRestantes == 0) {
        println("\n==Você não tem mais tiros. Fim do jogo!==")
        println("Total de pontos: $pontos")
    }

    println("\nVisão das jogadas no Tabuleiro")
    imprimirTabuleiro(tabJogador)

    println("\n==Visão dos navios no Tabuleiro==")
    println(" ✈️ Porta Aviões")
    println(" 🚢 Cruzador")
    println(" 🚤 Rebocador")
    imprimirTabuleiro(arrayTabuleiro)

}
jogarTurno()
println("\nDeseja Jogar novamente?")
val resposta = readln().trim()
```

```
        if (resposta == "sim"||resposta == "Sim") {
            continue
        } else {
            println("Como você não digitou Sim o jogo irá finalizar...")
            break
        }
    }

}
```