//// 1 ////

```
fun main()
{
   print("Enter day number: ");
   var day_number : Int = readln().toInt();
   when (day_number)
   {
      1 -> println("Sunday");
      2 -> println("Monday");
      3 -> println("Tuesday");
      4 -> println("Wednesday");
      5 -> println("Thursday");
      6 -> println("Friday");
      7 -> println("Saturday");
      else -> println("Out of range");
   }
}
```

//// 2 ////

```
fun main() {
   println("Enter triangle sides lengths: ");
   print("A = "); var A : Int = readln().toInt();
   print("B = "); var B : Int = readln().toInt();
   print("C = "); var C : Int = readln().toInt();

   var temp : Int;
   if (A < B) { temp = A; A = B; B = temp; }
   if (A < C) { temp = A; A = C; C = temp; }
   temp = triangle[0]*triangle[0] - triangle[1]*triangle[1] - triangle[2]*triangle[2];

   when {
      temp == 0 -> println("Triangle is right");
      temp > 0 -> println("Triangle is obtuse");
      else -> println("Triangle is acute");    }
}
```

//// 3 ////

```
fun main()
{
   print("Enter mark: ");
   var Mark : Int = readln().toInt();
   print(when (Mark) {
      5 -> "Excelent";
      4 -> "Good";
      3 -> "Credit";
      2 -> "Failure";
      1 -> "Terribly";
      else -> when { Mark > 5 -> "Imposible"; else -> "EXPELLED!"} });
}
```

//// 4 //// – ВЫНЕСЕНО НА ОТДЕЛЬНЫЙ ЛИСТ НИЖЕ

//// 5 ////

```
fun main() { var inp = readln().toInt(); print(if (inp > 0) «+» else if (inp < 0) «-« else «0»); }
```

//// 6 ////

```
import kotlin.random.Random
fun main() {
   var Num : Int = Random.nextInt(0,100);
   var Input : Int;
   var Attempts : Int = 0;
   println("Guess the number");
   while (true) {
      Attempts++;  print("- ");
      Input = readln().toInt();
      if (Input == Num) break;
      if (Input > Num) println("Smaller");
      else println("Bigger");
   }
   println("Victory! $Attempts attempts");
}
```

```
//// -- 4 -- ////
fun main() {
    print("Enter time in format HH:MM - ");
    val Time = readlnOrNull()?.trim() ?: return println("Incorrect input");

    if (!Regex("""^\d{1,2}:\d{2}$""").matches(Time)) return println("Incorrect format");

    val (h, m) = Time.split(":").map { it.toIntOrNull() ?: return println("Incorrect time values"); }

    if (h !in 0..23 || m !in 0..59) return println("Time out of range");

    val hoursText = numberToText(h) + " hour" + if (h != 1) "s" else "";
    val minutesText = if (m > 0) numberToText(m) + " minute" + if (m != 1) "s " else " " else "";
    val suffix = if (h < 12) "after midnight" else "past noon";

    if (h + m == 0) println("Midnight"); else println("$hoursText $minutesText$suffix");
}

fun numberToText(num: Int): String {
    val units = listOf("", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine");
    val teens = listOf("Ten", "Eleven", "Twelve", "Thirteen", "Fourteen",
                       "Fifteen", "Sixteen", "Seventeen", "Eighteen", "Nineteen");
    val tens = listOf("", "", "Twenty", "Thirty", "Forty", "Fifty");

    return when {
        num == 0 -> "Zero";
        num < 10 -> units[num];
        num < 20 -> teens[num - 10];
        else -> tens[num / 10] + if (num % 10 != 0) " " + units[num % 10] else "";
    }
}

//// 7 ////     //// 9 ////   // Задания дублируются //
fun main() { println(readln().count(); }
```

```
//// 8 ////
fun main() {
   println("Select food classification:" + "\n1 - By culinary category" + "\n2 - By cooking method");
   var Category : Boolean? = when (readlnOrNull()?.trim() ?: "")
   { "1" -> true; "2" -> false; else -> null; };
   if (Category == null) return print("Incorrect variant");
   if (Category) {
      println("1 - Appetizers and salads" + "\n2 - First courses" + "\n3 - Second courses" +
            "\n4 - Side dishes" + "\n5 - Baked goods and desserts");

      print(when (readlnOrNull()?.trim() ?: "") {
         "1" -> print("Cooking time: 5 - 20 min");
         "2" -> print("Cooking time: 20 - 90 min");
         "3" -> print("Cooking time: 15 - 60 min");
         "4" -> print("Cooking time: 10 - 40 min");
         "5" -> print("Cooking time: 30 - 120 min");
         else -> return print("Incorrect vatiant"); };
   }
   Else {
      println("1 - No cooking (fruits, nuts, yogurt)" +
            "\n2 - Cooking (porridge, soups, eggs, pasta)" +
            "\n3 - Frying (cutlets, steak, pancakes, potatoes)" +
            "\n4 - Baking (pies, meat, vegetables in the oven)" +
            "\n5 - Stewing (goulash, pilaf, stew)" +
            "\n6 - Grill/barbecue (shashlik, steaks, fish)" +
            "\n7 - Fermentation/marinating (kimchi, sauerkraut, dried meat)");
      print(when (readlnOrNull()?.trim() ?: "") {
         "1" -> Cooking time: 0 - 5 min"; "2" -> "Cooking time: 5 - 60 min";
         "3" -> "Cooking time: 5 - 40 min"; "4" -> "Cooking time: 20 - 120 min";
         "5" -> "Cooking time: 40 - 180 min"; "6" -> "Cooking time: 10 - 60 min";
         "7" -> "Cooking time: 1 day - several weeks"; else -> return "Incorrect vatiant"; } );
   } }
```

```
//// 10 ////
fun main()
{
    print("Select payment method: " +
            "\n1|a - Cash" +
            "\n2|b - Card or NFC" +
            "\n3|c - PayPal" +
            "\n4|d - Bank account" +
            "\n5|e - QR-Code" +
            "\n>>> ");

    val PaymentType : String = readlnOrNull()?.trim() ?: return println("Incorrect input");
    var PaymentTypeChar : Int = PaymentType[0].toInt() - 49;
    if (PaymentTypeChar > 5) PaymentTypeChar -= 48;

    print(when (PaymentTypeChar)
            {
        0 -> "Put cash in the bill acceptor";
        1 -> "Put your card into card receiver or attach to ATM";
        2 -> "Enter your PayPal account and password";
        3 -> "Enter your bank account and password";
        4 -> "Scan QR-code using bank app on your phone";
        else -> "Incorrect variant";
    });
}
```

```
//// 11 //// // Чистотой кода пришлось пожертвовать, чтобы всё уместить //
fun main() {
    print("Enter blood group (O/A/B/AB): ");
    var blood_type : String? = readln()?.uppercase()?.trim();
    print("Enter rh factor (+/-): ");
    var rh_factor : String? = readln()?.trim();
    if (blood_type !in listOf("O", "A", "B", "AB") || rh_factor !in listOf("+", "-"))
        return println("Incorrect input");
    var recipients : List<String> = GetRecipients(blood_type!!, rh_factor!!);
    var donors : List<String> = GetDonors(blood_type, rh_factor);
    println("You can be recipient of groups:\n${recipients.joinToString(", ")}");
    println("You can be donor for groups:\n${donors.joinToString(", ")}");
}
fun GetRecipients(blood_type: String, rh_factor: String): List<String> =
    when (blood_type to rh_factor) {
        "O" to "-" -> listOf("O-, O+, A-, A+, B-, B+, AB-, AB+");
        "O" to "+" -> listOf("O+, A+, B+, AB+");
        "A" to "-" -> listOf("A-, A+, AB-, AB+");
        "A" to "+" -> listOf("A+, AB+");
        "B" to "-" -> listOf("B-, B+, AB-, AB+");
        "B" to "+" -> listOf("B+, AB+");
        "AB" to "-" -> listOf("AB-, AB+");
        "AB" to "+" -> listOf("AB+");
        else -> emptyList()};
fun GetDonors(blood_type: String, rh_factor: String): List<String> =
    when (blood_type to rh_factor) {
        "O" to "-" -> listOf("O-");
        "O" to "+" -> listOf("O-, O+");
        "A" to "-" -> listOf("O-, A-");
        "A" to "+" -> listOf("O-, O+, A-, A+");
        "B" to "-" -> listOf("O-, B-");
        "B" to "+" -> listOf("O-, O+, B-, B+");
        "AB" to "-" -> listOf("O-, A-, B-, AB-");
        "AB" to "+" -> listOf("O-, O+, A-, A+, B-, B+, AB-, AB+");
        else -> emptyList()};
```

```
//// 12 ////  // Аналогично с 11 – код сжат //
fun main() {
    val dict: Map<String, Country> = mapOf(
    "USA" to Country(9.8e6, 330e6, "Washington", "USD", "UTC 5-10", 26e12),
    "China" to Country(9.6e6, 1410e6, "Beijing", "CNY", "UTC+8", 18e12),
    "India" to Country(3.3e6, 1420e6, "New Delhi", "INR", "UTC+5:30", 3.7e12),
    "Russia" to Country(17.1e6, 144e6, "Moscow", "RUB", "UTC+2 to +12", 2.3e12),
    "Brazil" to Country(8.5e6, 214e6, "Brasília", "BRL", "UTC-2 to -5", 2e12),
    "Germany" to Country(357_000.0, 83e6, "Berlin", "EUR", "UTC+1", 4.3e12),
    "France" to Country(551_000.0, 68e6, "Paris", "EUR", "UTC+1", 3e12),
    "UK" to Country(243_000.0, 67e6, "London", "GBP", "UTC 0", 3.1e12),
    "Japan" to Country(378_000.0, 125e6, "Tokyo", "JPY", "UTC+9", 4.3e12),
    "Canada" to Country(9.98e6, 39e6, "Ottawa", "CAD", "UTC-3.5 to -8", 2.1e12),
    "Italy" to Country(301_000.0, 59e6, "Rome", "EUR", "UTC+1", 2.1e12),
    "South Korea" to Country(100_000.0, 52e6, "Seoul", "KRW", "UTC+9", 1.8e12),
    "Mexico" to Country(1.96e6, 127e6, "Mexico City", "MXN", "UTC-6 to -8", 1.8e12),
    "Australia" to Country(7.68e6, 26e6, "Canberra", "AUD", "UTC+8 to +10", 1.7e12),
    "Turkey" to Country(783_000.0, 85e6, "Ankara", "TRY", "UTC+3", 1.1e12));

    print("Enter Country name: ")
    val Input : String = readlnOrNull()?.trim() ?: return println("Incorrect input");
    var max_similarity_coef : Double = 0.0;
    var current_similarity_coef : Double = 0.0;
    var current_country : String = "";
    for (country in dict.keys) {
        current_similarity_coef = dice_coefficient(country, Input)
        if (current_similarity_coef > max_similarity_coef)
        { max_similarity_coef = current_similarity_coef; current_country = country; }
    } println();
    if (max_similarity_coef < 0.15) println("Country \"$Input\" not found");
    else if (max_similarity_coef == 1.0) dict[current_country]?.print_info();
    else {
        println("Country \"$Input\" not found. \nMay be you meen \"$current_country\"?\n");
        println("$current_country info:\n");
        dict[current_country]?.print_info(); } }
```

```
data class Country(
    val square: Double,
    val population: Double,
    val capital: String,
    val currency: String,
    val timeZone: String,
    val gdp: Double)
{
    fun print_info()
    {
        print("Square = $square km²!" +
                "\nPopulation ≈ $population" +
                "\nCountry: $capital" +
                "\nCurrency: $currency" +
                "\nTime zone: $timeZone" +
                "\nGDP $gdp");
    }
}

fun dice_coefficient(s1: String, s2: String): Double {
    if (s1.isEmpty() || s2.isEmpty()) return 0.0;
    if (s1 == s2) return 1.0;

    fun get_bigrams(s: String): Set<String> = s.windowed(2).toSet();

    val bigrams1 : Set<String> = get_bigrams(s1.lowercase());
    val bigrams2 : Set<String> = get_bigrams(s2.lowercase());

    return (2.0 * bigrams1.intersect(bigrams2).size) /
            (bigrams1.size + bigrams2.size);
}
```

//// 13 ////

```
fun main() {
    val HttpErrors : Map<Int, HttpError> = mapOf(
    100 to HttpError("Continue", "Запрос принят, клиент должен продолжать отправку данных."),
    101 to HttpError("Switching Protocols", "Сервер принимает смену протокола, запрошенную клиентом."),
    102 to HttpError("Processing", "Запрос обработан, но ответ ещё не сформирован."),
    200 to HttpError("OK", "Запрос успешно выполнен."),
    201 to HttpError("Created", "Ресурс успешно создан."),
    202 to HttpError("Accepted", "Запрос принят, но обработка не завершена."),
    204 to HttpError("No Content", "Запрос выполнен, но содержимого нет."),
    300 to HttpError("Multiple Choices", "Запрос может привести к разным результатам."),
    301 to HttpError("Moved Permanently", "Ресурс перемещён навсегда."),
    302 to HttpError("Found", "Ресурс временно доступен по другому URL."),
    304 to HttpError("Not Modified", "Содержимое не изменилось с момента последнего запроса."),
    400 to HttpError("Bad Request", "Некорректный запрос."),
    401 to HttpError("Unauthorized", "Необходима авторизация."),
    403 to HttpError("Forbidden", "Доступ запрещён."),
    404 to HttpError("Not Found", "Ресурс не найден."),
    405 to HttpError("Method Not Allowed", "Метод запроса не поддерживается."),
    408 to HttpError("Request Timeout", "Время ожидания запроса истекло."),
    500 to HttpError("Internal Server Error", "Ошибка на стороне сервера."),
    501 to HttpError("Not Implemented", "Сервер не поддерживает запрошенный метод."),
    502 to HttpError("Bad Gateway", "Сервер получил некорректный ответ от другого сервера."),
    503 to HttpError("Service Unavailable", "Сервис временно недоступен."),
    504 to HttpError("Gateway Timeout", "Тайм-аут при ожидании ответа от другого сервера."),
    505 to HttpError("HTTP Version Not Supported", "Версия HTTP не поддерживается сервером.")  );

    print("Enter error code: ");
    val ErrorCode : Int = readlnOrNull()?.trim()?.toIntOrNull() ?: return println("Incorrect input");

    if (ErrorCode !in HttpErrors.keys) return println("Error code not found");
    println("Code: $ErrorCode – ${HttpErrors[ErrorCode]!!.name}: ${HttpErrors[ErrorCode]!!.description}");
}

data class HttpError(val name: String, val description: String);
```