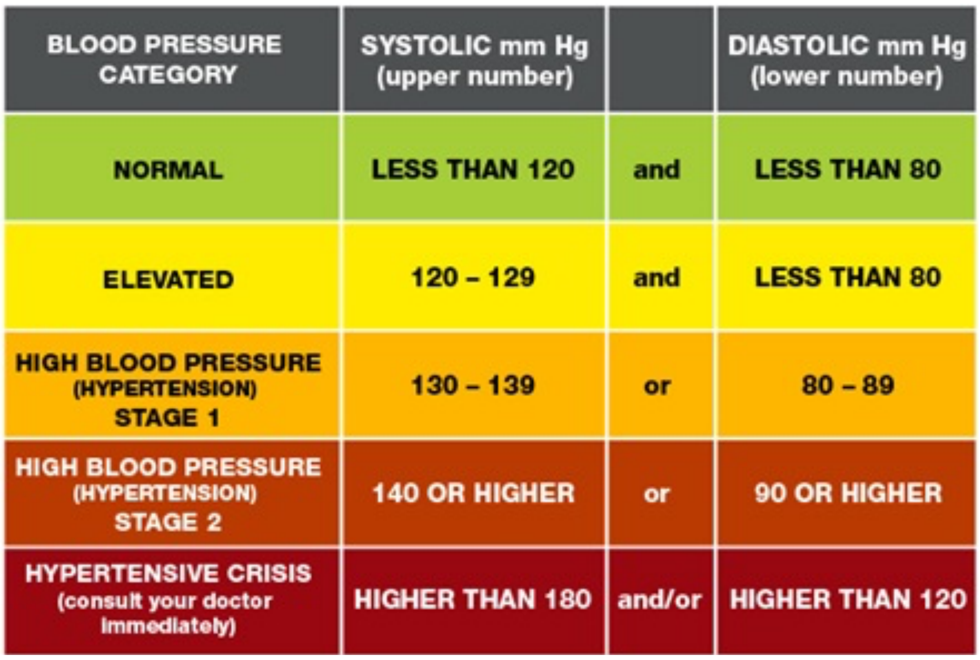
Exercise 1

Write a small program which generates pairs of systolic and diastolic blood pressure values and prints a textual interpretation of the values, according to the following table:



BLOOD PRESSURE CATEGORY	SYSTOLIC mm Hg (upper number)		DIASTOLIC mm Hg (lower number)
NORMAL	LESS THAN 120	and	LESS THAN 80
ELEVATED	120 – 129	and	LESS THAN 80
HIGH BLOOD PRESSURE (HYPERTENSION) STAGE 1	130 – 139	or	80 – 89
HIGH BLOOD PRESSURE (HYPERTENSION) STAGE 2	140 OR HIGHER	or	90 OR HIGHER
HYPERTENSIVE CRISIS (consult your doctor immediately)	HIGHER THAN 180	and/or	HIGHER THAN 120

Exercise 2

Please model the followings:

A bank can issue debit and credit cards.

Cards in general have the following operations:

- Pay (amount) -> [success/failure] – when card is used directly in payment transactions
- Withdraw (amount) -> [success/failure] – withdraw money from an ATM
- Deposit (amount) – deposit money to an ATM

Debit cards only work if balance is > 0.

Credit cards are allowed to access a credit line in a limit specific to each card instance.

Credit cards reward payment actions (not withdrawals) with some bonus, in the percentage of the amount payed. The percentage is global, according to the bank's actual benefit plan.

Exercise 3

Write a Kotlin function which calculates the Factorial of a positive integer number N.
Example: N = 5 -> 5! = 1*2*3*4*5 = 120

Exercise 4

Make the ‘render’ function more concise from the below snippet:

```kotlin
data class Rectangle(val x: Int, val y: Int, val w: Int, val h: Int)

class Paint {
    var color: Long = 0x00FF00
    var strokeWidth: Int = 5
    fun drawRectangle(rect: Rectangle) {
        println("Drawing $rect color: $color stroke: $strokeWidth")
    }
}

fun render(paint: Paint?, rectangles: List<Rectangle?>) {
    if (paint != null) {
        paint.color = 0xFF0000
        for (rect in rectangles) {
            if (rect != null) {
                paint.drawRectangle(rect)
            }
        }
    }
}
```

Exercise 5

Given the following data structure:

```kotlin
data class HeartRateEntry(val date: Long, val value: Int)

fun populateData(vararg dataPair: Pair<Long, Int>): List<HeartRateEntry> =
    dataPair.map { HeartRateEntry(it.first, it.second) }

val data = populateData(
    1612310400L to 76,
    1612310400L to 89,
    1612310400L to 44,
    1612224000L to 47,
    1612224000L to 115,
    1612224000L to 76,
    1612224000L to 87,
    1612137600L to 90,
    1612137600L to 167)
```

1. Print the minimum heart rate value

2. Print the average heart rate value
3. Print all heart rate values above 100
4. Print heart rate values grouped into a map, where keys are the dates, and values are lists of heart rate values measured on a specific date (represented by the key)
5. Print maximum heart rate values per each day