Exercise lecture arrays and functions: The (dancing) Alarm-clock Robot

In this exercise you will build an alarm-clock robot using functions and arrays. As a lot of our students have difficulties to get out of bed in the morning, we would like you to create an alarm robot that alternates a blinking led with a sounding buzzer. You do this as follows:

- Start by examining how you can control the led and buzzer of your robot.

- Once you know this, try to find out how to let your led and buzzer blink and sound for a predefined number of milliseconds.
Hint: have a look at the delay() function.

- Now build a function that lights up the led for a predetermined number of milliseconds. The function should take as parameter an integer that represents the number of milliseconds the led should light up. Do the same for the buzzer. After having defined such a function, implement a new function that alternates the blinking and sounding of the led and buzzer for a predetermined number of seconds. This function has three parameters: one to determine how long the alternation of led and buzzer should last, one that determines how long the led should blink each time and one that determines how long the buzzer should sound each time. So your "alternating" function should use the two previous functions to let the led blink and make the buzzer sound.

- We now almost have all the basic building blocks for our alarm clock robot. Still, we need to define a few other data-structures. More precisely, we want an alarm clock that can be set differently for each of the five weekdays. To keep things simple, we won't work with a real clock but just simulate these timings. Therefore you should declare two arrays that contain the timings for each day. More specifically, the first array represents for each day in how many seconds the alarm should go off. The second array says for each day when the alarm should stop. The following drawing represents what is meant:

	day 1	day2	day3	day4	day5
array1	2	3	2	1	3
array2	6	7	8	5	9

This means that if we iterate over these arrays the following should happen: For day 1 the alarm has to start in 2 seconds and should stop in 6 seconds (starting from 0 and not from 2). This means the alarm should go off for 4 seconds. Next, day 2 will start in 3 seconds and the alarm has to go of for four seconds. And so on. Implement the array data-structures to represent this mechanism. You can initialize them as in the example given, or write a function that lets a user set these timings.

- Finally implement a loop that represents the alarm for a week. Therefore it should iterate over our two arrays, and blinking and sounding the buzzer for the determined number of seconds for each day.

- If this was all pretty straightforward for you and you have time left, then we have a final challenge for you: Please make the robot dance (for instance move in a circle) each time the alarm goes off.

Exercise lecture object oriented programming: Kitchen Recipes

In chapter 8 of the textbook the author introduces the example class "buritoRecipe" illustrating the concept of OO programming. In this exercise you will extend this example. More precisely you have to design the classes "cakeRecipe" and "pancakesRecipe" analogous to the "buritoRecipe" class. Of course these should be added in such a way that you use the strength of OO modelling, i.e. the concept of inheritance. Doing it like that it will be easy to add other recipes as well in the future. Sketch the class structure you will use, and then implement this in the processing environment.