Exercise 0  Rate-Monotonic Scheduling

Given a fixed number $N$ of periodic tasks we want to implement a rate monotonic scheduler.

- Have a look at the `task.h` headerfile that contains the basic task structure and some utility functions.

- In order to schedule our tasks we need to find the hyper-period $h$ of the tasks. The hyper-period is given by the least common multiple $lcm()$ of the task periods $p_i$:

$$h = lcm(p_0, \ldots, p_{N-1}). \quad (1)$$

The least common multiple of two positive numbers $a, b$ can be calculated using the greatest common divisor $gcd$:

$$lcm(a, b) = \frac{a \cdot b}{gcd(a, b)}. \quad (2)$$

In the header `util.h` an implementation of the $gcd$ can be found. Implement a function `task_hyperperiod` that takes a task array (pointer) and the number of tasks and returns the hyperperiod. Use the following rules for the calculation of the $lcm$:

$$lcm(a) = a \quad (3)$$

$$lcm(a, b, c) = lcm(a, lcm(b, c)). \quad (4)$$

- Given the hyper-period $h$ and the $N$ tasks we can start implementing our rate monotonic scheduler for one hyperperiod. In the file `rmscheduler.h` you find some skeleton code and some hints for the implementation. The function `task_sort_rate` sorts the tasks according to the period in ascending order. Remember that for the rate monotonic scheduler always the task with the highest rate $r_i = \frac{1}{p_i}$ is selected. After a task has finished execution it can not be selected for scheduling until it becomes ready again at its period time. If a running task gets interrupted it still needs to execute for its remaining time at some point. Check if the schedule is feasible and the deadlines are met (here the deadline is the period).

- Use the following $N = 3$ tasks $T_1 = (4, 1), T_2 = (5, 2), T_3 = (20, 5)$ with a hyper-period of 20 to test your implementation together with the example schedule in the figure below.