Exercise 1 (Load Balancing):
In the exercises we presented the problem of load balancing for identical machines and permanent jobs. We proved the theorem that the GREEDY-algorithm (assigning the current job to the machine with smallest load before the assignment) is \((2 - \frac{1}{m})\)-competitive, where \(m\) is the number of machines.
Prove that the upper bound shown there is also valid in the case of jobs with finite duration (whether known or unknown).

(20 points)

Exercise 2 (Scheduling):
In the following we are looking at a scheduling problem where \(n\) jobs \(j_1, \ldots, j_n\) arrive. There are the following limitations:

- there is only one machine \(M\)
- \(M\) can work on exactly 1 job at any time
- each job \(j_i\) must be executed on \(M\) continuously for a period of \(p_i > 0\)
- each job \(j_i\) can be started at the release time \(r_i\) or later
- each job \(j_i\) has a deadline \(d_i\)

The goal is to devise a schedule \(\Sigma\) (i.e., to assign the jobs to \(M\)) such that the maximum delay is minimized over all jobs. The schedule determines the start times \(s_i\) for all jobs. The delay \(l_i\) of a job is defined as the difference between \(c_i\), the time the job is finished (depends on the chosen heuristic), and the deadline:

\[
l_i(\Sigma) = c_i(\Sigma) - d_i
\]  

Hence, the maximum delay is \(L_{\text{max}} = \max_{1 \leq i \leq n} l_i\).
Note that in this model the maximum delay may become negative. However, non-positive deadlines are unrealistic.

Another model uses delivery times as follows: Each job has a delivery time \( q_i \). When the job has been completed, it is delivered only after an additional time \( q_i \) (for example on an extra machine). Now, different delivery times may overlap. For a job \( j_i \), the value \( s_i + p_i + q_i \) denotes the end of the delivery (for the first model above this would give us \( q_i = -d_i \)). Let \( L_{\text{max}}^* \) be the smallest maximal delay over all schedules. Then it holds that

\[
L_{\text{max}} = \max_{1 \leq i \leq n} s_i + p_i + q_i
\]

(2)

Let further \( l_i = c_i + q_i \) be the end of the delivery for job \( j_i \), then it holds that

\[
L_{\text{max}}^* \geq P = \sum_{i=1}^{n} p_i
\]

(3)

\[
L_{\text{max}}^* \geq r_i + p_i + q_i
\]

(4)

Consider Graham’s algorithm LIST SCHEDULING (LS): When (a machine) \( M \) is free, assign to it the first available job. A job is available after it has been released.

(a) Why are the bounds (3) and (4) valid?

(b) Prove that: \( L_{\text{max}}^{LS} < 2L_{\text{max}}^* \)

(10 + 20 points)

Exercise 3 (DOUBLE COVERAGE algorithm for \( k \)-server problem):
We have seen in Exercise 2.2 that the GREEDY-algorithm for the \( k \)-server problem is not necessarily competitive. Let’s look at the following algorithm for \( k \) servers on a line:

**DOUBLE COVERAGE**

- If the request lies outside the convex hull of all servers, move the closest server to serve the request.

- Else the request lies between two servers. Move both of them - with the same velocity - towards the request, until (at least) one server reaches the request point.

Reconsider the worst-case example of Exercise 2.2 (3 request points on a line). Why does the DOUBLE COVERAGE algorithm not produce an arbitrarily bad result in this instance? (10 points)